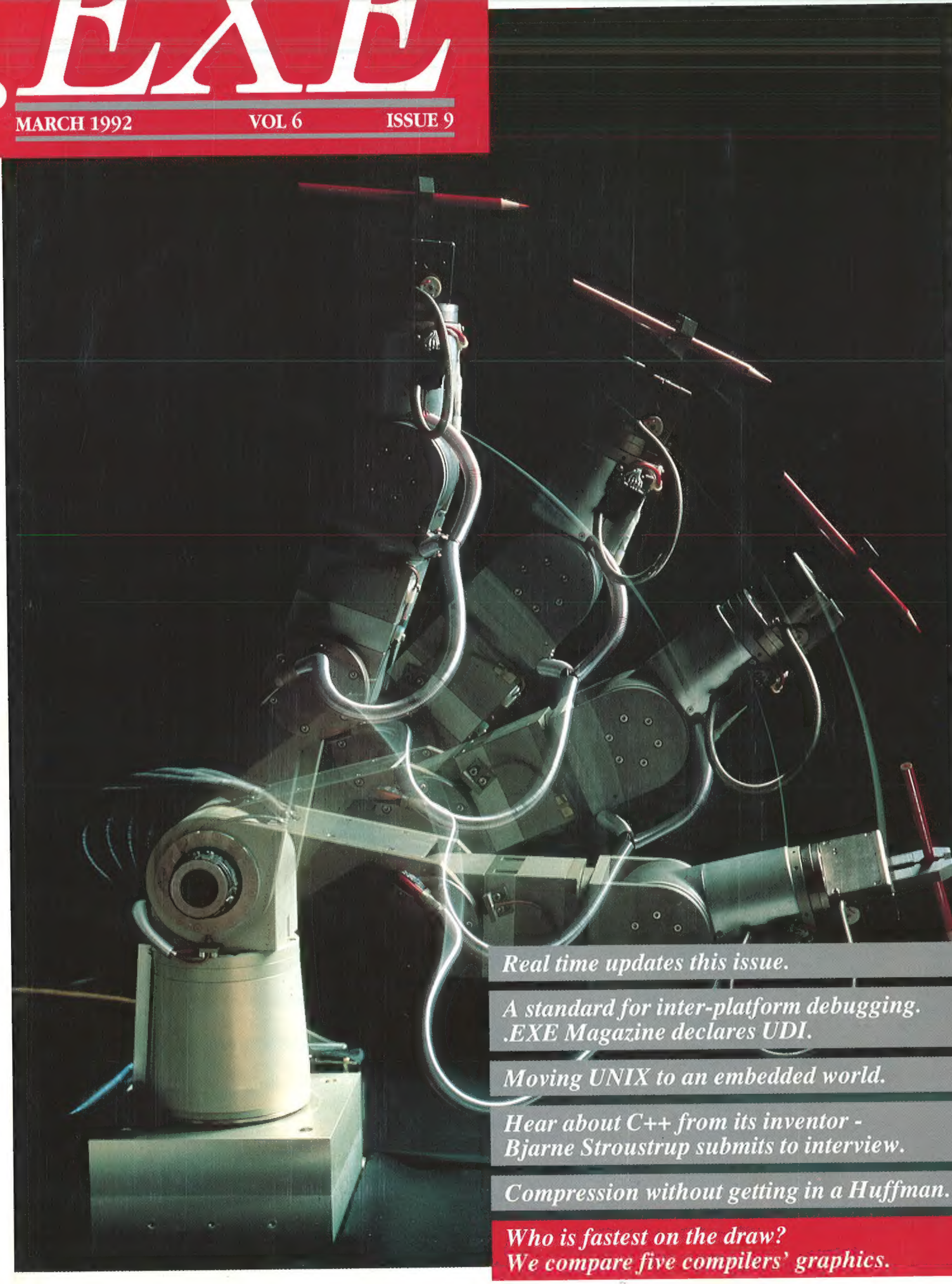
£3.20

.EXE

MARCH 1992 VOL 6 ISSUE 9

The Software Developers' Magazine



Real time updates this issue.

A standard for inter-platform debugging. .EXE Magazine declares UDI.

Moving UNIX to an embedded world.

Hear about C++ from its inventor - Bjarne Stroustrup submits to interview.

Compression without getting in a Huffman.

Who is fastest on the draw? We compare five compilers' graphics.

THE TRAINING LEADERS

The best path for complete C training begins at QA

C has become the dominant language in computer programming and provides access to the software industries major application program interfaces.

C is probably now, or shortly will be, your way forward in program development – and QA can really help you along that path.

Only QA's extensive C curriculum totally supports the training needs of all C programmers, from beginner through to highly experienced professional. Each course can stand alone or can be the entry point to reach any level of skill or knowledge you may set yourself.

In common with all QA courses, our instructors are experienced professionals themselves, our training materials are perfected for efficient learning, and all our own skill and energy is channelled towards helping you get the maximum benefit from your training.

Send for more details today or call Samantha Trinder on 0285 655888.

BEGINNERS START HERE

C Primer
2 days
Introduction to HLL concepts via C • An overview of the C programming environment • Developing simple C programs • Data typing, block structure, score concepts • Introducing C functions and modular programming • How to use the C library and preprocessor • Working with larger programs • Introduction to advanced techniques

OTHERS START HERE

C Programming
4 days
C program structure and style • Data types and variables • Flow of control in C programs • Pointers and address manipulation • Portability and efficiency considerations • Debugging C programs • Working with larger C programs

OR ANYWHERE

Advanced C
4 days
C review • Common problems • ANSI features • Advanced pointers • Standard library • Dynamic memory techniques • Efficiency and optimisation

ELSE

C++ Programming
4 days
C++ today • Declaring and using classes • Operator and function overloading • C++ advanced syntax concepts • Single and multiple inheritance • Performance considerations • Class libraries and application frameworks • OOP: the benefits and pitfalls

ALSO

PASCAL

Pascal Programming
4 days
Structured programming style • Data types, variable declarations • Statements and expressions • Decision and repetition in Pascal • Arrays, records and files • Pascal development tools and libraries • Common extensions

MASM

Masm Programming
4 days
8088/86/286 architecture • MASM program structure and style • Interrupt handling • 8088/86/286 instructions • Advanced segmentation • Modular program development • Interfacing to high level languages • 80286/386 overview

Mail to QA Training Ltd, Cecily Hill Castle, Cirencester, Gloucestershire GL7 2EF, UK.

Tick here for details:

TRAINING: ☐ C ☐ Pascal ☐ Masm ☐ Object-Oriented ☐ Networks ☐ UNIX/AIX ☐ OS/2 ☐ Windows

☐ CONSULTANCY

☐ SOFTWARE DEVELOPMENT PRODUCTS

NAME ____________ TITLE ____________

COMPANY ____________

ADDRESS ____________

TEL. NO. ____________

AEX01

QA TRAINING

C Programming Training For All Levels

CIRCLE NO. 535

Vol 6 Issue 9

.EXE

March 1992

General

.EXE Magazine is independent and not affiliated to any vendor of hardware, software or services. It is published by Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH.
Tel: (Advertising/Editorial/Production) 081 994 6477
(Subscriptions) 0442 824501
Facsimile: 081 994 1533 Telex: 8811418 SPACES G
ISSN: 0268-6872

Subscriptions

.EXE Magazine is a monthly journal for software developers. It is available only by subscription, at a cost of £35 per annum (11 issues) in the UK, £50 for 2 years. Overseas subscriptions cost £45 per annum, £70 for 2 years. 'A subscription implies that this journal will be sent to the subscriber until one of the three expires' - AG Macdonell. The magazine is published around the 26th of the month preceding the cover date. There is no January issue. To subscribe or obtain details of overseas/academic rates, please call 044282 4501, or write to The Subscriptions Manager, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We can invoice your company or take ACCESS and VISA cards. In case of a query about your subscription, please call 044282 4501.
Back issues are available at a cost of £3.50 each - call our Chiswick office (081 994 6477) for a list of issues still in print.

Editorial

Editorial enquiries should be addressed to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We welcome letters, opinions, suggestions and articles from our readers. Please write for a copy of our Contributors' Guide.
Information contained in .EXE is believed to be correct. If errors are found, we will endeavour to publish a clarification in the next available issue.
From time to time, we offer to copy the PC program code described in an article onto diskette for our readers. In this case, please send a blank, formatted disk with a self-addressed, prepaid mailer to the editorial address given above. We can copy both 5.25" and 3.5" disks.
The publishers can accept no liability for any consequences of using software distributed in this way.

Advertising

If you are interested in advertising in this magazine, please write to the Display Advertising Manager/Recruitment Sales Manager (as appropriate), .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH, or call 081 994 6477 for details of our advertising rates.

Pronunciation

.EXE Magazine rhymes with 'not sexy magazine'.

Copyright

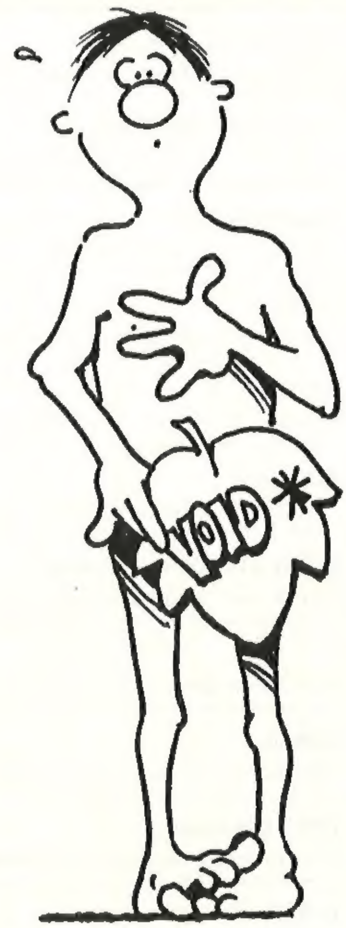
Issue theme: Developing for Real Time

Censoring C++

Paul Kemp's no prude but he finds C++ classes in need of a fig leaf.

I have a problem with private parts. Well, those of C++ classes to be precise. The problem is partly aesthetic and partly practical. Let's start with the cosmetics. If the behaviour and usage of a C++ object is entirely defined by its `public` and `protected` interface, then why is its `private` data so unashamedly exposed in the class declaration? The internal implementation of a class is (or should be) of no interest to the user of that class. This is one of the most fundamental concepts in object-oriented programming - yes, that old chestnut data abstraction or encapsulation. Not only is it unnecessary to see `private` data and methods, but their presence in a class declaration can be confusing and obscure the use of the object, especially if there are a lot of them.

The practical difficulties lie in the fact that the 'visibility' of this `private` data compromises the maintainability of application code. It is part of a wider problem associated with developing and maintaining large C++ systems. To be fair, the problem is partially to do with the current set of development tools available to C++ programmers. Large applications have many classes with complex and far-reaching interdependencies. Changes in the design of classes (a *very* frequent occurrence in the initial stages of development) will inevitably necessitate the recompilation of source code.



Unfortunately, because many classes are dependent on other classes this can lead to a domino effect, whereby a minor change to a class declaration can leave the programmer staring blankly at his screen while half of his application code is recompiled. What C++ programmers obviously need is an intelligent `make` utility capable of figuring out what needs to be recompiled and what doesn't. Working this out for yourself and hand coding the appropriate `make` file is about as exciting as using a CASE tool.

Design changes that affect the `public` interface of a class are pretty fundamental and the programmer will have to accept the consequences, but `private` data is implementation-specific and should not affect the *users* of a class (the client code). Unhappily this is not the case. If the visible `private` data of a library class is modified (for example, because the *implementation* of a `public` member function has changed) then all of the application code which uses that class must be recompiled. So much for data abstraction. During the suck-it-and-see design phase of a C++ project the problem of cascading recompilations can become quite intolerable.

Faced with these problems on a sizeable project (and being far too ~~lazy~~ pressed for time to do the complicated `make` file), a colleague and I developed a simple but effective technique. Undoubtedly variations of it have been considered and used by other C++ developers. Like many programming techniques, it is a trade-off between elegance and efficiency. In brief, the 'visible' `private` data of all classes is reduced to a single, abstract `void` pointer. This in fact points to a `struct` which contains the *real* `private` data items. In the header file, for example:

```
class obj
{
private:
  void * pData;
public:
      obj( int data );
      ~obj();
  int GetData();
};
```

The `struct` containing the real data is defined in the source file which implements the class's methods, and only has scope within that source file. The object's constructor allocates memory for this structure and saves the pointer in `pData`. Member functions that wish to access any item of `private` data must declare a pointer to the real data `struct` and cast `pData` to the appropriate type. Hence in the source file:

```
struct Imp
{
  int val;
};
obj::obj( int data )
{
  Imp * p = new Imp;
  pData = (void *)p;
  p->val = data;
}
obj::~obj()
{
  Imp * p = (Imp *)pData;
  delete p;
}
int obj::GetData()
{
  Imp * p = (Imp *)pData;
  return p->val;
}
```

When using this technique, `private` data (contained within the `Imp` structure) can be changed at will without any modifications to the header file, thus avoiding knock-on recompilations.

Of course there are overheads and restrictions. Class definitions must be kept in a separate header file from their source and you can't have any inline functions which access the real `private` data. When you do wish to access the class' `private` data items, there is the overhead of an additional level of indirection (`p->`). The constructor also has the overhead of dynamically allocating memory for the data items on the heap. Remember a destructor must also be provided which frees this memory.

The technique doesn't hide `private` member functions and is probably only appropriate for complex classes, but it can help C++ developers to avoid the 'domino effect'. So cover up your private parts - modesty is sometimes the best policy.

Special thanks to Peter Sabine-Bacon whose bright idea this was.

Actor 4.0

Version 4.0 of Whitewater's thespian OOP language incorporates database-independent SQL libraries and DLLs for Paradox, Excel, SQL Server, dBASE, DB2, Oracle and OS/2 Extended Edition. Improvements have also been made in the ObjectWindows class library. Plain vanilla Actor 4.0 costs £129 or £375 for the full-blown Professional development system. Neow markets the product in the UK on 0628 668334.

New Home for FG

Zortech's Flash Graphics library has been revitalised, thanks to a new company called FlashTek. As a result of Symantec's acquisition of Zortech in September 1991, Joe Huffman, the original author of FG, has modified the library so that it can be used with the Borland C++ compiler. There are also plans to release a BGI interface to the Flash Graphics engine which will enable applications written using BGI to be re-compiled with FG. Flash Graphics is produced by Flashtek on 0476 74108 and costs £110.

Deal on BC++

With the imminent release of Microsoft C V7.0, Borland has introduced a special offer on the Borland C++ V3.0 with Application Framework package. In the same vein as Microsoft, Borland is tempting users of Microsoft C, Quick C, Zortech C++ and JPI with an attractive upgrade price of £189.95 (RRP of BC++ with Application Framework is £439.95). Borland can be contacted on 0734 320022.

Parallel C++

Packaged as an add-on to 3L's Parallel C compiler, Parallel C++ is a cfront-based implementation of AT&T's C++ V2.1 language specification. Designed for developing transputer-hosted applications, Parallel C++ costs £295 and requires a copy of Parallel C. Users who don't already have 3L's C compiler can get both products bundled for £795. 3L is on 0506 415959.

HP graphics

Workstation Source has introduced HT BASIC V3.3, a compiler which enables programs written using HP Rocky Mountain BASIC to be compiled and run under DOS. The new version supports loadable device drivers for keyboard, display and HPGL hardcopy devices. A 32-bit, VCPI and DPMI compatible version of the compiler is also available. HT BASIC for DOS costs £495. The 32-bit HT BASIC 386 costs £695. Workstation Source is on 0628 75252.

HP's USL C++

Unix System Laboratories announced that it will be using the C++ exception handling technology developed by Hewlett-Packard in its forthcoming USL C++ Language System. Exception handling is also featured in HP's own new C++ compiler (HP C++ V3.0). HP has stated that its new C++ compiler provides '...the first exception handling facilities for a C++ compiler'.

'We are offering C++ V3.0 on the heels of USL's announcement of its C++ Language System Release 3.0,' added Richard Owen (HP's Workstation Marketing Manager), 'making us the first vendor to put some of these key new technologies into developers' hands.'

The lack of exception handling is generally acknowledged to be an important flaw in the present Release 3.0 C++ programming model. HP C++ V3.0 will be available in the second quarter of 1992 for the HP9000 series 700 workstations at a cost of £1,950 for a one-seat licence. Hewlett-Packard is on 0344 369369.

Soft Disk Mirror

Recovery is a LAN utility which provides disk-mirroring in software. It can be run from any workstation and works by continually analysing the source and destination drives. Files that have been modified since the last time the mirror was performed are copied from the source drive (which may be any logical drive including virtual disks, floppy disks and hard disks) to the destination mirror drive. If a file is busy it is picked up the next time through the mirror process which continues until all the files have been copied (or until it is stopped by the user). It is possible to specify a datestamp for copying only the most recent files.

During disk-mirroring, Recovery provides the user with various information including a list of files that have been copied and a list of files remaining. Recovery is priced at $99 and is distributed by Nonstop Networks Ltd on 0101 212 4818488.

Enhanced Empress

The Empress Database Server is the latest addition to the Empress RDBMS. It achieves increased performance by utilising the Internet protocol for communicating over LANs, instead of relying on an underlying filing system such as NFS.

Database server is able to take advantage of multi-processor hardware and, using a Global Data Dictionary, it enables transparent access to distributed databases. It provides a server spooler which accepts database requests from clients for one or more databases. Each request is assigned a sub-process which may be executed on a separate CPU. The sub-process accesses the Empress RDBMS and returns information directly to the client.

Database Server is included with the latest version of the Empress RDBMS which costs £7000 for the Sun2 developer's edition. Empress Software can be reached on 0483 861990.

E-Mail DLL

Finansa Ltd has announced that its new WinMail V1.0 product is the first E-Mail package to be distributed as a DLL. WinMail supports Novell's Message Handling Service V1.1 which provides compatibility with over 100 gateways including Fax, Telex, Pager and mail from IBM, UNIX and Macintosh etc. Incoming mail is automatically put into categorised folders, based on sender etc. An Address Book is provided and WinMail can be configured to delete junk-mail automatically. The WinMail API is also included. A five-user version of WinMail V1.0 is priced at £395 (10-user version costs £695). For more information contact Finansa on 0582 662268.

Multi-user PC

Technology Concepts Ltd has developed PCC/MX, a multiple serial port device that lets up to eight users/terminals connect to a central PC. Unlike standard serial port cards, the PCC/MX consists of two parts. Internally, the PC is fitted with the PCC/SYNC processor card containing a 16 MHz 80186 processor and 512 KB of RAM. The processor card is used to coördinate the activities of two PCC/MXs. Each PCC/MX consists of eight V.24/RS232 serial ports and a parallel printer port. Technology Concepts claims that the PCC/MX comms line 'runs at speeds up to 1MHz', and, '...individual terminals can be sited up to 30m from it'.

The PCC/MX also offers remote connections using high speed modems and four units can be linked together, supporting a total of 32 users.

Device drivers are available for several operating systems including XENIX, CDOS, PC MOS and OS/9000. The PCC/MX costs £1425 and includes the PCC/SYN front-end processor card which supports two PCC/MX units. For more information contact Technology Concepts Ltd on 0633 872611.

GREAT PERFORMANCE GETS YOUR ATTENTION GREAT TECHNOLOGY KEEPS IT

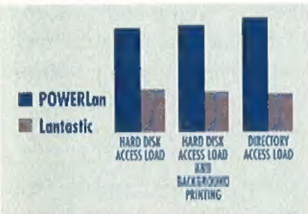
POWERLan

DOS Networking

"POWERLan swept all of the performance categories on our benchmark tests."

"If you need to link DOS, OS/2, Unix, and other platforms in an interoperable network ... POWERLan is the product to choose"

"you can't ignore this powerhouse, POWERLan's features speak for themselves; its speed convinces"



PC MAGAZINE EDITORS' CHOICE

June 25, 1991
POWERLan, Version 2.10

CIRCLE NO. 537

POWERsave

Network Backup

"This is the fastest single-file restore we found in any of the systems, including server-based units. We didn't believe it either until we ran the test three times."

"simple to use"

"Performance Technology may have the fastest workstation-based drive in Netwaredom."

"offers excellent progress tracking"

"operates ... at warp speed"

LAN Times, June 17, 1991

LAN TIMES RECOMMENDED 1991 FOR NETWORKING

POWERfusion

DOS - UNIX Connection

"In our benchmark tests, Performance Technology lived up to its name."

"... the POWERfusion family can support large networks of busy users and still deliver"

"If you're looking for a way to connect your Unix box to a network of DOS-based PCs, look no further."

PC MAGAZINE EDITORS' CHOICE

May 28, 1991
POWERfusion, Version 1.3

"Performance Technology is selling glue technology that can help solve a variety of local- and wide-area networking problems. Lord knows, that's something we all need."

BYTE, March 1991

Sure, we like to make products that go fast. But, performance is just one quality we take seriously. Operating in multi-vendor environments is high on our list too. And there's lots more. Call us to get the rest of the story.

POWERbridge

LAN to LAN Connection

"Usually I wonder about a product that has "power" associated with it, but POWERbridge ... deserves the honor."

"POWERbridge supports nine operating systems ... links all the networks you'll ever have."

LAN Times, February 4, 1991

"... well, you haven't lived until ... (you) ... hop through a NetWare LAN to an SMB server running under Unix" (with POWERbridge)

BYTE, March 1991

AUTOMATED OFFICE SYSTEMS
COMPUTER SOFTWARE DEVELOPMENT

C.M.S. SOFTWARE LTD

A. O. S. SOFTWARE DEVELOPMENT
First Floor, 46A Chiswick High Road,
Chiswick, London W4 1SZ
Telephone: 081 995 7716
Facsimile: 081 995 7865

PC NeXT

NeXT Computer Inc has unveiled NeXTstep 486, a version of the company's UNIXish object-oriented system software that runs on Intel 80486-based machines. NeXT's own machines are based on the powerful Motorola 68040 chip. NeXTstep 486 will be offered in two versions: a user version retailing at £695 and a developer version, price to be confirmed. Both will be distributed on CD-ROM media. System requirements are pretty hefty with 8-16 MB RAM and 400 MB disk space needed for the developer version. Contact NeXT for more info on 081 5650005.

C + C++ = ?

Supporters of C and C++ will soon be brought together in a new, all encompassing conference which has been given the tongue-twisting title of 'C Plus C++ in Action'. The conference will take place on the 8-12 June 1992 at the London Gatwick Airport Hilton. There will be over forty tutorials and discussions on such issues as Advanced C Language Topics, Transition from C to C++ and Object-Oriented Analysis and Design. Details of the conference can be obtained from the Boston University Conference Office on 071 2592032.

Tool up

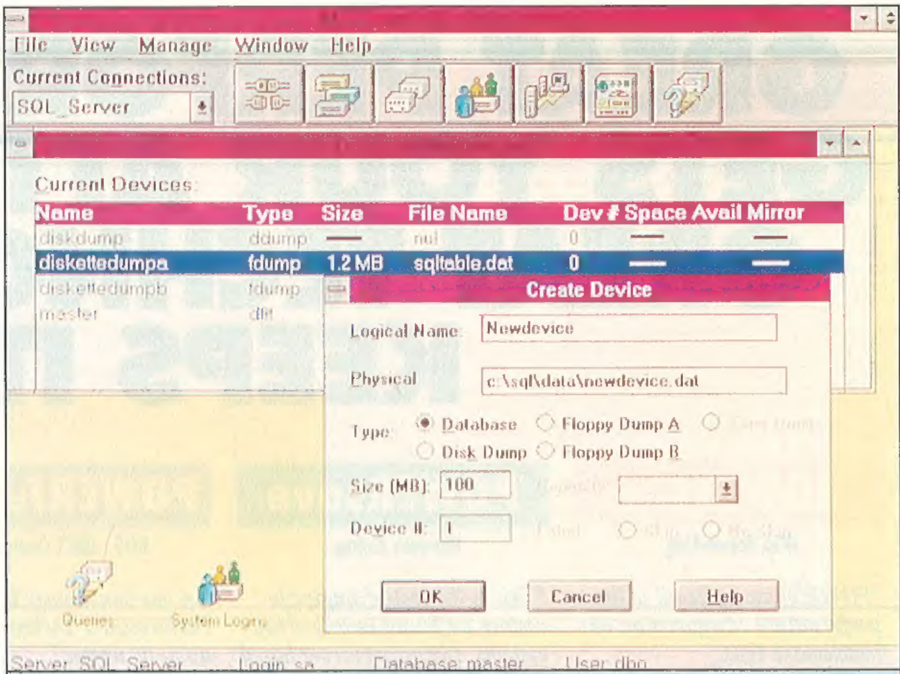
OOP Messiah Bertrand Meyer will be chairing the 7th International Conference & Exhibition of Technology of Object-Oriented Languages and Systems. TOOLS Europe '92 will be held in Westfalenhallen Dortmund, Germany from 30 March - 2 April, 1992. Phone: 010 33 45325880, Fax: 010 33 45325881 or Email: `tools@eiffel.fr`.

CodeBase offer

Get a free copy of Borland's Turbo C++ compiler and 'World of C++' video when you purchase both Sequiter Software's CodeTranslator V1.0 (RRP £129) and CodeBase V4.5 (RRP £255) from The Software Construction Company at a combined cost of £385. The Software Construction Company is on 0763 244114.

310 ASM progs

The latest release of the ASM Utility Library from EMS contains a total of 310 shareware programs distributed on eight 1.44 MB diskettes and requires over 25 MB of disk space. EMS provides a handy lookup program which help you to find a given utility quickly. The ASM library covers many subject areas including BIOS, Compression, Networks and TSRs. The ASM Utility Library costs $59.50. Contact EMS professional Shareware Libraries on 0101 301 9243594.



ADVANCED BUSINESS GRAPHICS

Giant leap for SQL Server

Microsoft's strategic client/server database SQL Server has leapfrogged in version number from the current V1.11 to V4.2.

The renumbering reflects the inclusion of all core features found in Sybase versions of the product for UNIX and VMS platforms. Microsoft has stated that the numbering is consistent with Sybase and represents the commitment by Microsoft and Sybase to deliver the same features in all versions of SQL Server.

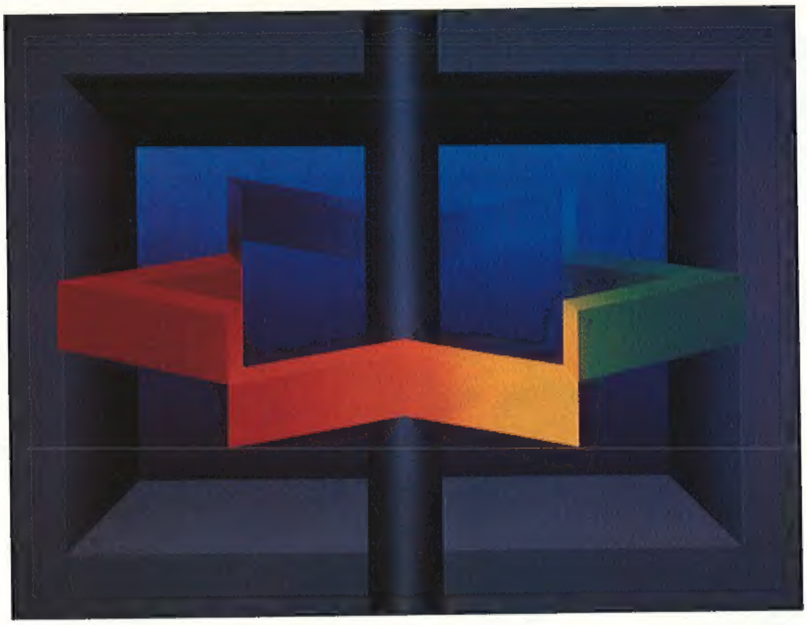
New facilities in V4.2 include database device mirroring for fault tolerance, cascading triggers, online dynamic tape backup, placement of tables on specific devices and increases in overall system capacity. A Windows-hosted server administration tool (see picture) is also included which should simplify configuration, tuning and backup of SQL Server systems. The most significant enhancement for programmers is the inclusion of support for scrollable database cursors. Cursor support greatly simplifies the development of data browsing applications with the ability to scroll backwards and forwards through row-level data. Positioned updates and deletes may also be performed.

Also included in the box are a hardware-independent version of OS/2 and the Open Data Services (ODS) toolkit. ODS allows the development of SQL Server gateways and extensions to interface with disparate data sources on other platforms such as VMS, UNIX, MVS and VM. ODS applications can be triggered by remote stored procedure calls from SQL Server.

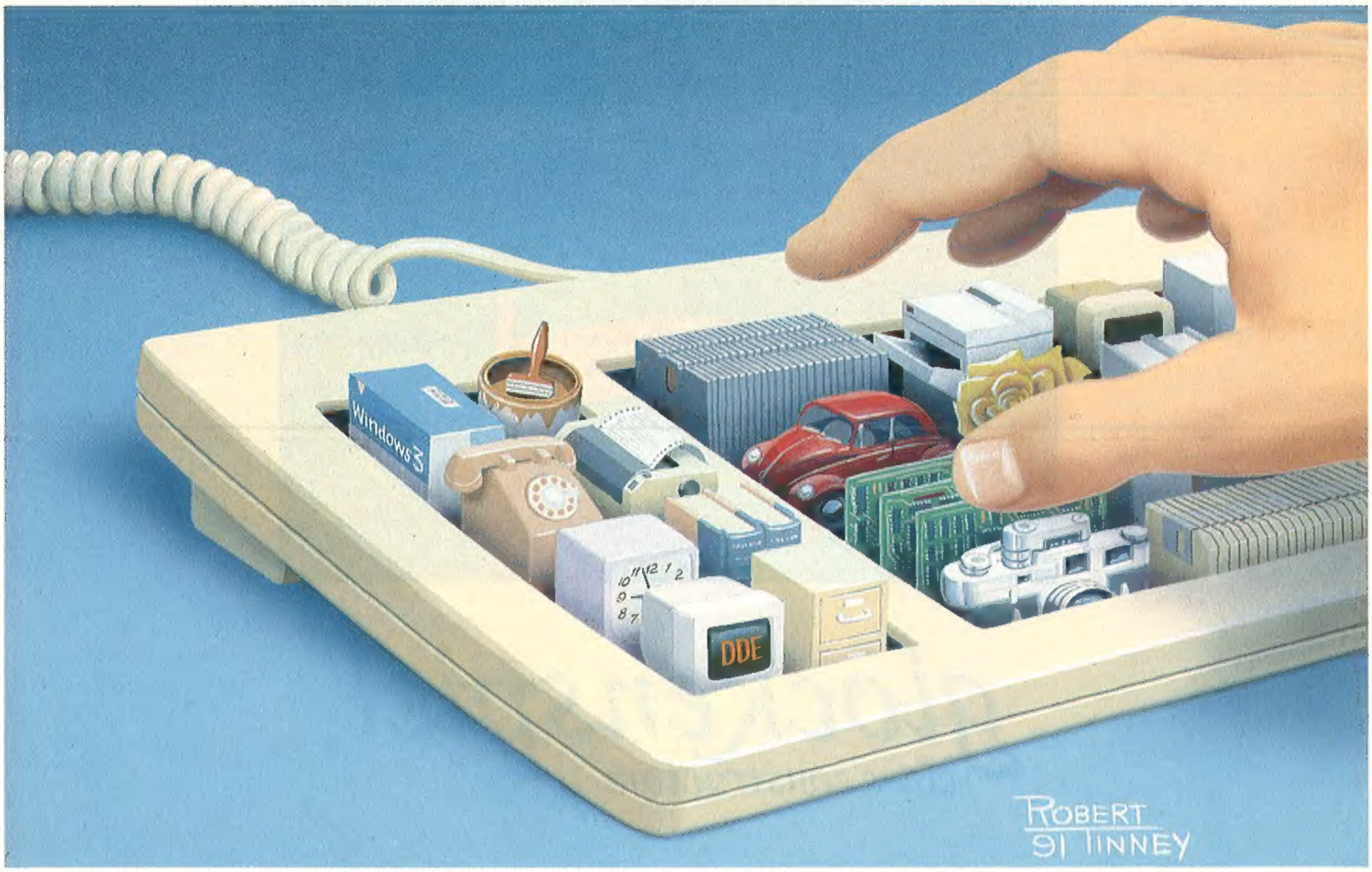
The connectivity of SQL Server has also been extended with the availability of Gateway Link/Gateway Services from SQL Solutions, a subsidiary of Sybase, and Microsoft's SQL Bridge. Gateway Link is built using ODS technology and allows SQL Server front-end applications to access VMS and UNIX-based minicomputer databases like Oracle, Ingres, Informix and VAX RMS. SQL Bridge is a two-way network gateway that connects Microsoft SQL Server and Sybase SQL Server environments, allowing clients and servers to communicate across different networks. This means that UNIX, Macintosh and VMS clients can now access data on Microsoft's OS/2-based version of SQL Server.

SQL Server and LAN Manager are Microsoft's only remaining ties with the operating system *non grata* OS/2. While a Windows NT version of SQL Server is in the pipeline, it is still fairly embryonic, and Microsoft has stated that it will continue to support OS/2 as a client and server platform. Support for OS/2 2.0 clients has been promised but Microsoft was non-committal on the subject of whether SQL Server would be revamped to take advantage of OS/2 2.0's 32-bit API.

SQL Server V4.2 prices start at £2,295 for a 10-user server pack with a free upgrade for existing users of V1.11. SQL Bridge costs £1,995. Gateway Link is available from SQL Solutions for £3,000 and retail prices for the various Gateway Services range from £15,640 to £100,000 depending on the hardware platform and the number of users. SQL Solutions is on 0344 360101 and Microsoft is on 0734 270000.



Objects At Your Fingertips.



Now, if you want to develop applications for Windows 3.0, there's a fast and easier way to do it with the premier object-oriented programming language. Smalltalk/V.®

With Smalltalk/V Windows, you can explore, prototype, build finished applications and ship them runtime free.

You can tap into applications using DDE so effortlessly you don't have to be a Windows expert to do it.

And with one of the world's most comprehensive class libraries, you can choose our objects or easily build your own.

But whatever you develop, it will be portable between the Windows, OS/2 and Mac versions of Smalltalk/V.

With so much at their fingertips, more people are solving more problems with Smalltalk/V than any other object-oriented programming system.

At only £330 + VAT and no runtime charges, you can solve them, too.

Just call us at 071-436 9481. And see why programming Windows has never been easier.

Smalltalk/V Windows

COCKING & DRURY (SOFTWARE) LTD

180 Tottenham Court Road, London W1P 9LE Phone 071-436 9481 FAX 071-436 0524

CIRCLE NO. 539

Instant GUI

XVT Inc has released XVT-Design V1.0, an interactive design tool that lets the developer create the visual side of a GUI without having to resort to hard-coding in a high-level language.

XVT-Design works in conjunction with the XVT GUI library to enable the development of platform-independent GUI applications on Macintosh, Microsoft Windows, Presentation Manager, Open Look, and OSF/Motif systems. Using XVT's Universal Resource Language, resources generated with XVT-Design on one platform can be loaded onto any other XVT platform.

XVT-Design lets the developer set up menus and 'draw' controls onto a window or a dialog box. Default 'File' and 'Edit' menus are provided. Unlike a resource editor, once the GUI has been created, XVT-Design will generate a framework C program, including header files and a makefile. The MS-Windows/Mac versions of XVT-Design cost £595 each. XVT-Design is distributed in the UK by Personal Workstations on 071 231 0333.

DIY FoxPro

Ohio-based Fox Software has begun shipping its Library Construction Kit (LCK) for FoxPro 2.0.

The package provides tools to create external libraries of C and assembler routines that can be integrated into FoxPro in its interactive, application or executable forms. A cut-down version of Watcom C V8.5 is included in the pack. This was necessary because FoxPro itself is written in Watcom C. There is also a deal whereby purchasers of the LCK can can upgrade to to full Watcom C V8.5 package which includes Watcom's debugger, profiler, protected-mode compiler and linker, graphics library, Windows and OS/2 versions of the C run-time library, floating-point Emulator, MAKE utility, object code librarian and additional documentation.

The company has also incorporated support for Novell NetWare's Transaction Tracking System into V2.0 of FoxPro/LAN, its LAN-based version of FoxPro. This is in conjunction with a new pricing structure aimed at capturing a larger share of the xBASE market.

FoxPro/LAN 2.0 now retails for £795 for six concurrent users on one server. The Library Construction Kit costs £345. Contact Fox Software in the UK on 0462 421999 for more details.

Micro Wars

The battle for supremacy in the hotly contested 386/486 market has taken a new turn. Both Intel and AMD have launched new 386 microprocessors.

Intel has announced a cheaper version of the Intel386 SL. This 20MHz cache-less microprocessor is targeted at the portable PC market and offers on chip power management.

AMD has introduced a number of revved-up devices including a 33MHz Am386SXL and a 40MHz Am386DXL microprocessor. AMD is also planning an offensive on Intel's monopoly of the 486 arena later this year. The Am486DX will be available in 25, 33 and 50 Mhz versions and will offer reduced power consumption. There will also be a 25 MHz Am486SX.

Intel's 20 MHz Intel386 SL costs £60. The 33 MHz Am386SXL costs $76 (=£42) and the 40 MHz Am386DXL costs $114 (=£63).

Intel can be reached on 0793 696000. AMD is on 0483 740440.

New from SCO

The Santa Cruz Operation (SCO) has been busy upgrading its product line. SCO UNIX SVR3.2 V4.0 is the latest release of the popular UNIX operating system for 386/486 machines. The new version can be obtained on CD-ROM and includes SCO Shell (a menu-driven user interface to the UNIX system) and provides support for loadable device drivers. It is now able to access up to 512 MB of RAM and can cope with disk drives larger than 1.2 GB. The V4.0 development system features new tools, updated libraries and certified compliance with the POSIX and XPG3 standards, along with compliance with Issue 2 of the Intel Binary Compatibility Specification (iBCS-2).

SCO UNIX MPX Release 2.0 is a new version of SCO's Multi-processor Extensions to the SCO UNIX operating system. MPX works in conjunction with SCO UNIX or SCO Open Desktop. In a multi-processor system, with UNIX running on the first processor, each copy of MPX enables another processor to share the overall workload of the system. MPX now supports hardware from several vendors including ALR, Compaq, NEC and Olivetti.

SCO UNIX V4.0 is available on diskette, tape and CD-ROM media at £495 for a two-user licence and £995 for an unlimited-user version. The development system costs £995. SCO MPX Release 2.0 is priced at £1,195. The Santa Cruz Operation can be contacted on 0923 816344.

AI in a box

Integral Solutions' PopLog, the program development environment, is now available for UNIX-based PCs. The PopLog toolkit integrates Prolog, Lisp and Pop-11. The toolkit includes PopLog Rules (for automatically generating expert-system rules), PopLog-Neural (an interactive Neural Network design tool) and PopLog-Flex (an expert system toolkit). PopLog costs £4,500. Integral Solutions Ltd is on 0256 822028.

Network Solutions

Novell is now offering a 5 and a 50 user version of the Netware V3.11 network operating system. Netware V3.11 incorporates 32-bit multi-tasking and supports DOS, Windows, Macintosh, OS/2 and UNIX platforms. Disk duplexing, read-after-write verification and disk mirroring are also provided. The prices of the 5 and 50 user versions of Netware V3.11 are £800 and £3700 respectively. Novell can be reached on 0344 860400.

Real Time C

The JMI C Executive real time kernel now supports MetaWare's C compiler and PharLap's assembler and linker for the 80386. This latest version (C Executive V2.4A) provides device and I/O drivers for a 386 PC motherboard. A system debugger, CE-View, is also available. The developer's version of C executive costs £1500. CE-DOSFile is priced at £750 and the CE-View debugger costs £300. C Executive is distributed in the UK by RTS on 0624 623841.

ISO Graphics for PCs

Developers using the Zortech C/C++ compiler family can now write device independent graphics applications using S-GKS V3.0, a new implementation of the ISO certified Graphics Kernel System (GKS) called S-GKS V3.0. S-GKS has many advantages over standard DOS graphics libraries and it enables developers to write graphics applications which are source code compatible over several platforms including UNIX and VAX/VMS. S-GKS costs £325. For more information contact Scientific Software Ltd on 0628 890011.

Glock on AIX

Dublin-based Glockenspiel is planning to publish the first open C++ toolset for the new IBM AIX Software Development Environment Workbench/6000. Glock C++ 3.0 is a preprocessor based on AT&T's C++ V3.0. The company is also working on a C++ 3.0 compiler for OS/2 2.0. C++ 3.0 on AIX will cost about £1,500. Glockenspiel is on 010 353 1733166.

Letters

We welcome short letters on any subject that is relevant to software development. Please write to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked 'Not for Publication', it will be considered for inclusion in this section.

VR, a Panacea?

Sir,

As the real world gets more hazardous and unpleasant in the craze to travel further and faster each year, so the benefits of spending at least part of our lives in fantasy have an increasing appeal. So I enjoyed your articles in the December issue and was led to speculate on a new solution to one of my pet problems. Could VR help delay the time when our assault on the environment stifles us all in noise and poisonous fumes?

Could the programming industry come to the rescue with VR packages to allow would-be Porsche owners to simulate the antisocial features of their craving by acting out their fantasies in a safe and environmentally friendly manner? How about a really green Golf GTI comprising a bog standard box with a 1300cc engine for the essential journeys and a VR kit, for use only in the home, to satisfy that urge to drive faster and more noisily than everyone else?

Of course usage of the real and virtual kit must be mutually exclusive, and I can see the need to simulate some virtual spectators, perhaps a passenger or two... otherwise what would be the point? Could VR trendies be persuaded to display stickers on their back windows 'My VR car's a Bugatti' instead of the engine size and performance badges we see on the backs of so many cars today? Apart from the savings in cash, energy and lives, no more incentive to steal cars and race them around the streets when we drive nothing more powerful than a Ford Pop or Metro.

The possibilities are endless. Rush hours could become distant memories when we do our virtual work in virtual offices, and that Mr VR, what a nice boss, pleasure to do *his* bidding, (shame about the pay!). And just think how much slimmer we would be with virtual meals, more delicious than any mother made.

So go to it programmers! Britain may have lost out in designing and building real products but there is a whole virtual world out there where we can show how environmentally friendly we are! No more whinging about a solution looking for a problem!

Terry Smith
Leeds

Borland's secret bug-list?

Sir,

Being a relatively experienced C++ user, I was more than interested in the new Borland C++ compiler (V3.0) as reviewed in your December '91 issue. I am currently involved in a large ongoing client-server project using C++ on a Windows 3 platform, and our current compiler is Borland C++ V2.0. We upgraded to Borland 3.0 immediately, being keen to enjoy the extra features (such as a DPMI compliant tool set) and development tools included. As you may expect, we did some preliminary testing of the new compiler before unleashing it on the entire project. During this we found three obvious bugs in the package, which we believe should have been caught at 'alpha test' stage (these are not to be confused with the incompatibilities between Windows 3.1 and the IDE and other tools). The bugs range in seriousness from 'cosmetic' to 'very serious'. They are:

The Windows debugger does not return the correct termination code of a program (Borland tells me that it gets confused about the stack),

`return` statements in in-line constructors are flagged as syntax errors despite being perfectly valid syntax (and good practice some would say),

`return` statements in destructors trigger a bug in code generation, whereby base class destructors don't get called, so objects are not fully 'destructed'.

The first two bugs can be considered annoying, but cause no major problems. The third bug however is subtle and causes serious problems. No warnings are generated, and programs with the fault generally do not die immediately, they just fail to clean up certain objects properly. The overall impression is of a lack of thorough testing, of a product rushed to market to appear before the release of Microsoft C 7.0.

Borland Technical Support has now acknowledged these as bugs (after I sent them very small programs to demonstrate the bugs), and says that they will be fixed in future releases of the compiler. The problem that I have is that Borland will not tell me of any other serious compiler bugs, they will not ship a fixed version of the compiler (they say it may take six months), and they even denied that my test programs were valid C++ syntax until I quoted to them chapter and verse of *Ellis & Stroustrup*. I am left with a compiler with some known bugs, which should have been caught a long time before the product hit the shelves, and a suspicion that many more bugs may have slipped through the apparently inadequate testing, and which Borland may be aware of but won't admit to.

I am sending this letter to bring these bugs to the attention of any other users of this compiler, and in the hope that they will in turn tell others of any bugs that they find. Borland says that they will not release bug lists 'because no other PC software makers do'. Well, every copy of Zortech C++ contains a list of known bugs, and those recently fixed. Microsoft has been known (on demand) to release bug lists, as has Lotus. Until such time as software manufacturers show the maturity to release lists of known and serious bugs regularly, especially in development tools such as compilers, PC development will still be seen as a 'toy environment' compared to the open nature of serious and more stable worlds of UNIX *et al.*

Tim Meadowcroft
Cambridge

Zortech lib revisited

Sir,

In .EXE March 1991, I wrote a review of the Zortech Database Library which was not exactly complimentary.

In reviewing a new product, it is inevitable that first impressions tend to skew the balance, particularly where there are highly visible user-interface glitches.

Just as I am sure that writers of OSs receive few compliments, perhaps the best a database designer can hope for is that nobody notices it's there. For the record, I feel I should point out that since writing the review I have used the Zortech Database Library in two substantial applications and have found it to run flawlessly.

Finally, I haven't liked recent .EXE covers - it's getting to look more and more like a comic book. Maybe it doesn't matter because the magazine is not for retail sale, but I do on occasion point out articles to clients. I would urge you to go for a more professional look without selling out to the men in grey suits.

John Cant
PHD Computer Consultants
Edinburgh

Mr Cant is to be praised for coming forward with his revised opinion. Let this be an awful warning to all .EXE's contributors, as well as its Editor, of the danger of shallow reviews. The problem is, of course, that it is dangerous printing a review based on five months' heavy use of a package as a) you the readers are not all prepared to wait that long, and will attempt to find the info elsewhere and b) the actual product will have itself been superseded. However, the issue is not straightforward. Should we be printing more (or only) 'long term' reviews, as favoured by some other magazines? Your comments welcome.

Meanwhile, apologies are clearly due to Zortech in general and in particular to Steve Teale, who is the author of the Zortech database.

As for the .EXE's covers, perhaps Mr Cant has found the current offering and its predecessor more to his taste - Ed.

Bombed

Sir,

In the Security Supplement to February's .EXE Magazine, I presented an algorithm for defending against viruses. This consisted of a series of tests on the files as well as some simple tests on the system itself.

It turns out that some clones of MS-DOS (and even MS-DOS V5.0) can exhibit behaviour that would, under the circumstances described in my article, be considered suspicious. MS-DOS V5.0, for example, will map memory one paragraph short of the 640 KB limit when DOS=HIGH,UMB is set in CONFIG.SYS, and some DOS clones can map memory to as much as 1 KB short of the end of BIOS memory. Digital Research DOS (DR-DOS) V6.0 will load interrupts into the 64 KB above the Intel 1 Megabyte limit in 286 or better machines and use a JMP FAR instruction to access them.

Accordingly, I have enclosed updated files SYSCHECK.C and DOSMCB.C to account for these possibilities and have sent you Stealth Bomber version 2.2 to replace version 2.0 for distribution. Special thanks to Mark Hamilton for his time and expertise.

Kevin Dean
Toronto, Canada

We only sent out about four disks, all before Feb 6th, with the faulty version of STEALTH. If you were one of the recipients of these disks (you can tell because there is no mention of the update in the README.1ST file) please call the Editorial Office and we will mail you the corrected version. We'd like to echo Kevin's thanks to Mark Hamilton for his considerable help in this matter - Ed.

Letter of the Month

The writer of the best letter of the month, as judged by the Editor, will be receive a £20 book voucher, courtesy of Just Computer Books. The best letter is the one printed first. Please note that letters submitted to this page may be edited.

'Twixt Host and Target

The design of embedded system debuggers has, so far, been at the whim of the writer. A new interface standard will benefit both manufacturers and users, says Daniel Mann.

Code development for an embedded processor is generally more costly than development of code of equivalent complexity intended for execution on a 'general purpose' machine, such as a workstation. The embedded application code cannot benefit from an underlying support operating system such as UNIX. In some cases, developers may chose to first install a small debug support monitor or third-party executive which can offer a somewhat improved development environment. In the process of getting an embedded support monitor running or developing application code to run directly on the processor, emulation hardware may be employed. The availability of debug tools, and the extent to which it is possible to configure them, are important factors when selecting a processor for an embedded project.

The architecture of the latest RISC processors may be simplified compared to their CISC predecessors, but the complexity of controlling the processor operation has not been reduced. The use of register stacks and instruction delay slots and other performance enhancing techniques has lead to increased use of high-level programming languages such as C. The compiler has been given the responsibility of producing efficient assembly code, and the developer rarely deals with code which manipulates data at the processor register level. The increased productivity achievable by this approach is dependent on high-level debug support tools.

Developers of products containing embedded processors are looking to RISC for future products offering increased capability. The greater performance relative to RISC processor cost should make this possible. The suitability, cost and productivity of the tools available for code development are likely to be the major factor in deciding the direction ahead in preparing to tool-up for RISC.

This article describes a Universal Debug Interface (UDI) which is processor independent and enables greater debug tool configurability. A number of emulator and embedded monitor suppliers as well as high-level language debug tools suppliers are currently configuring their tools to comply with the proposed UDI standard. Current implementations are targeted for RISC processor code development. UDI should ease the choice in selecting tools and consequently selecting RISC. I shall concentrate on describing the Free Software Foundation's GDB C language source debugger's integration with UDI.

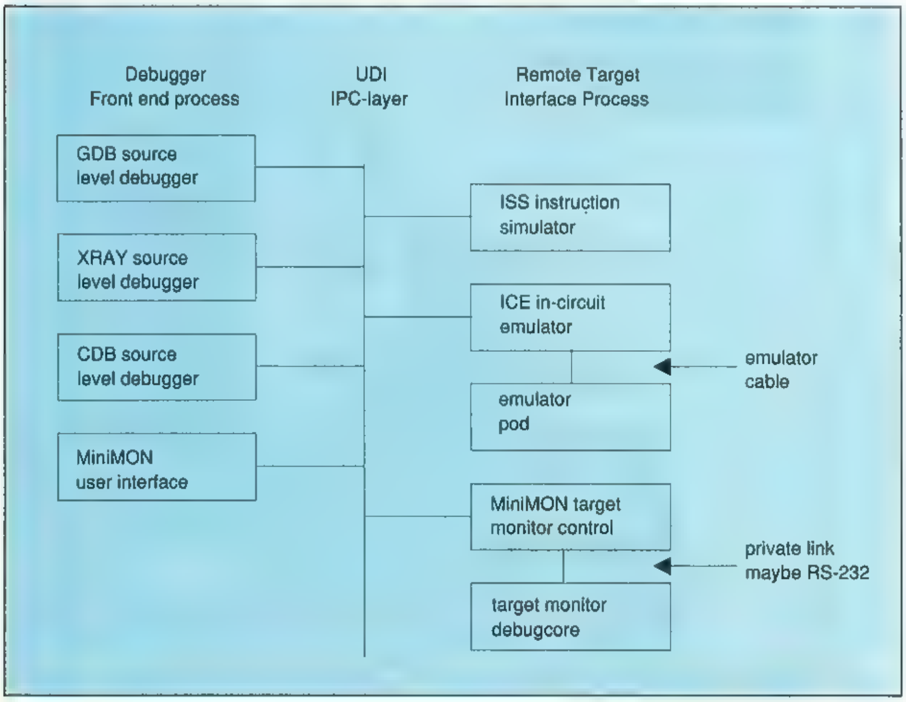


Figure 1 - Some UDI conforming debug tools

Tool Developers

A debug tool developer typically arranges for his product to be available for a range of popular processors. This normally means rebuilding the tool with the knowledge required to understand the peculiarities of each processor. If an enhancement is made to the debugger user interface, then normally the debugger source and the processor specific information must be recompiled and tested before customers are updated.

When developing code to run on a workstation, the processor supporting the debugger execution is the same processor running the program being developed. This means the debugger can make use of operating system services such as `ptrace()` (described later) to examine and control the program being debugged. When developing code for an embedded application, the program being developed is know as the *Target Program*, and executes on the *Target Processor* which is usually a different processor than the one supporting the debugger, known as the *Host Processor.* The host processor and target processor do not communicate via the

Nu-Mega
TECHNOLOGIES INC

Get Inside WINDOWS!

MICROSOFT WINDOWS
Version 3.0 Compatible Product

Debug Windows at the systems level!

Soft-ICE/W takes you inside Windows! Debug and explore with power and flexibility not found in any other Windows debugger! Soft-ICE/W allows you to debug at the systems or applications level or simply learn the inner workings of Windows.

- Debug VxD's, drivers and interrupt routines at source level
- Debug interactions between DOS T&SR's and Windows Apps
- Debug programs in DOS boxes
- Display valuable system information (from the total memory occupied by a Windows application, to the complex internal structures of Windows)

Soft-ICE/W uses the 386/486 architecture to provide break point capabilities that normally require external hardware. Nu-Mega, which pioneered this technology with the introduction of its award winning Soft/ICE for DOS, now gives Windows programmers the same debugging power... *and still at a software price.*

Own the debugger that combines the best "view" of Windows internals with the most powerful break points of any software debugger.

Soft-ICE/W . . . Only £259.00 +VAT

CodeView for Windows users: see what you're debugging without flash. **CV/1 version 2.0** runs CodeView in a graphics window while viewing your application screen. Runs on <u>any</u> display that supports Windows.

CV/1 . . . Only £95.00 +VAT

WHAT THE EXPERTS ARE SAYING

"Soft-ICE for Windows is great! It helped me find, in fifteen minutes, a killer bug in a Windows virtual device driver that had eluded two people for several months. I can't see doing Windows development of any kind -- whether writing Windows applications, device drivers, or even DOS programs that have to run under Windows -- without it. In addition to being great for finding bugs, Soft-ICE for Windows has been essential for my work on a forthcoming book: on Undocumented Windows. Soft-ICE for Windows goes anywhere and does everything, so it's essential for anyone who wants to poke around inside Windows Enhanced mode. DOS programmers will find it a perfect way to learn how the Windows DOS extender and DPMI server work, and how Windows interacts with DOS. Windows Enhanced mode is the hacker's paradise of the 90s, and Soft-ICE for Windows is the tool that every serious Windows or DOS hacker will need. Nu-Mega has done a brilliant job!"

Andrew Schulman
Software Engineer, Phar Lap Software
Editor, Undocumented DOS
Coauthor, Undocumented Windows (forthcoming)

Tel: (071) 833 1022
Fax: (071) 837 6411
3-5 Cynthia Street,
London N1 9JF

Now in the UK
System Science

RISK = NULL
30 DAY
MONEY-BACK GUARANTEE

MICROSOFT WINDOWS IS A REGISTERED TRADEMARK OF MICROSOFT CORP. Soft-ICE/W AND CV/1 ARE TRADEMARKS OF NU-MEGA TECHNOLOGIES, INC.

CIRCLE NO. 541

If you could create your own computer for developing software what would you build?

You'd need a system to increase your productivity, one that helped you create applications in half the time it takes you now – and complete projects ahead of schedule!

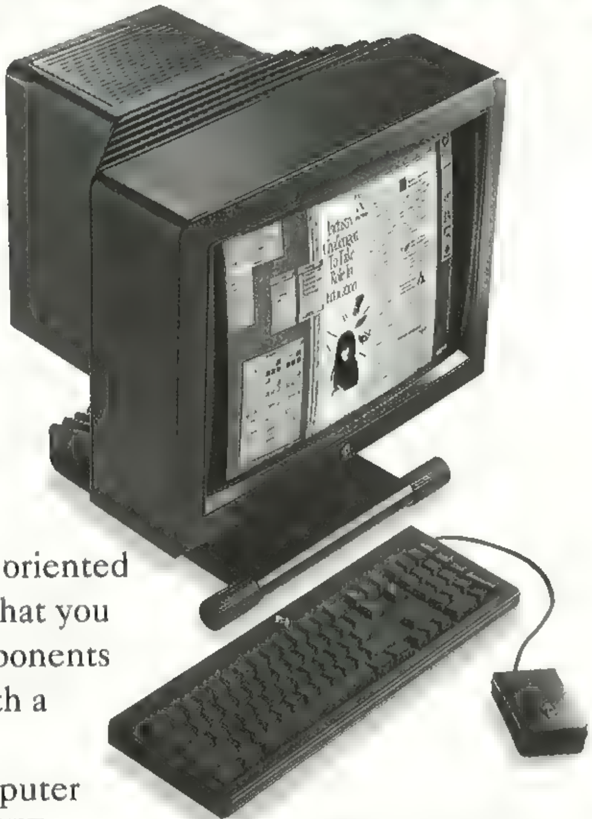
THE ULTIMATE DEVELOPMENT MACHINE.

You'd want a true multitasking system with virtual memory. Because that would give you the kind of computer that will let you compile a program in one window while debugging in another, so if your new application should crash your whole system won't die.

You'd give it plenty of power and speed to handle even the most complex projects, complete with UNIX and the Motorola 68040 processor.

With your users in mind, you'd want an Interface Builder so you could create an outstanding user interface – quickly and easily.

And you'd want an object oriented programming environment so that you could reuse your software components and it would help if it came with a complete object library.



Fortunately, there's a computer that already offers this, the NeXT computer. The tools provided by the NeXTstep operating environment make it easy to develop advanced, user-oriented programs.

So take the NeXTstep and call 081-565 0005 for more information, or fill out the coupon.

The Professional's Workstation

I'd like to receive more information on NeXT ☐
I'd like to receive more information on NeXT Developer Courses ☐

Please complete and return to: NeXT, 286 Bath Road, West Drayton, Middlesex UB7 0DQ.

Name:

Company:

Address:

County:

Telephone: Post code:

.EXE 3/92

CIRCLE NO. 542

```
/* Sample prototype of a UDI-p procedure */
UDIRead(
    /* source address on target */
  UDIResource     from,
    /* destination address on DFE host */
  UDIHOSTMemPtr   to,
    /* count of objects */
  UDICount        count,
    /* size of each object */
  size_t          size,
    /* count actual transferred */
  UDICount        *count_done,
    /* endian conversion flag */
  UDIBool         host_endian);
```

Figure 2 - UDI-p Library call

`ptrace()` system call, but via whatever hardware communication path links the two processors. The portion of the debugger which controls communication with the target processor is known as the target interface module, and whenever a change or addition is required in the communications mechanism, the debugger must be recompiled to produce a binary executable which is specific to the target-processor and target-communications requirements.

When the chipmakers turn out their latest whiz-bang RISC processor, the tool developer companies are faced with considerable development costs in ensuring their tools function with the new architecture. It is not uncommon for the availability of debug tools to lag a long way behind RISC chip introduction. Often tools are introduced with limited configuration options. For example, target processor communication may be according to a low-level debug monitor protocol, or an in-circuit emulator (ICE) protocol. Each debugger product has its own target interface module, this module must be developed for each debugger which wishes to communicate with the new target RISC processor.

An embedded application developer may have prior experience or a preference for a particular debug tool, but the only available communications path to the target may not be currently supported. This incompatibility may discourage the developer from choosing to use a new processor. It is therefore desirable that debuggers share communication modules and be more adaptable to available target processor interfaces.

Ideally a debugger from one company should be able to operate with, say, an emulator from another company. This would make it possible for a customer to select a little used debugger with a popular target monitor or vice versa.

The goal of the Universal Debug Interface (UDI) is to provide a standard interface between the debugger developer and the target communications module, so the two can be developed and supplied separately. In fact, an applications developer could construct his own communications module, for some special hardware communications link, as long as it complied with the standard.

UDI

If UDI were a specification at procedural level, then debugger developers and communication module developers would have to supply linkable images of their code, so the debug tool combination could be linked by the intended user. This is very undesirable because it would require a linked image for every tool combination. Additionally, the final linked program would be required to run on an single debug host. UDI actually relies on an inter-process communication (IPC) mechanism to connect two different processes. The debugger is linked into an executable program to run on the host processor - this process is known as the Debugger Front End (DFE). The communications module is linked as a separate process which runs on the same or a different host processor, and this process is known as the Target Interface Process (TIP). The two processes communicate via the UDI inter-process communication specification.

Two IPC mechanisms have so far been specified; one uses shared memory and is intended for DOS developers, the second uses sockets and is intended for UNIX and VMS developers. Of course, when the shared memory IPC implementation is used, the DFE and TIP processes must both execute on the same host processor. Using sockets with Internet domain communication enables the DFE and TIP to each execute on separate hosts on a computer network. Thus an applications developer can, from the workstation on his desk, debug a target processor which is connected to a network node located in a remote hardware lab. Using sockets with UNIX domain addresses (the method used to implement UNIX pipes) enables both processes to run on the same host.

Some of the currently available UDI conforming debug tools are presented in Figure 1. The inter-process communications layer defined by UDI enables the applications developer to select any front end tool (DFE) with any of the target control tools (TIP).

Because developers of UDI conforming tools must each have code which interfaces with the IPC mechanism according to the UDI protocol, the UDI community freely shares a library of code know as the UDI-p library. This code presents a procedural layer which hides the IPC implementation.

Consider the following example UDI-p call given in Figure 2. The DFE code calls the `UDIRead` function which *transports* the

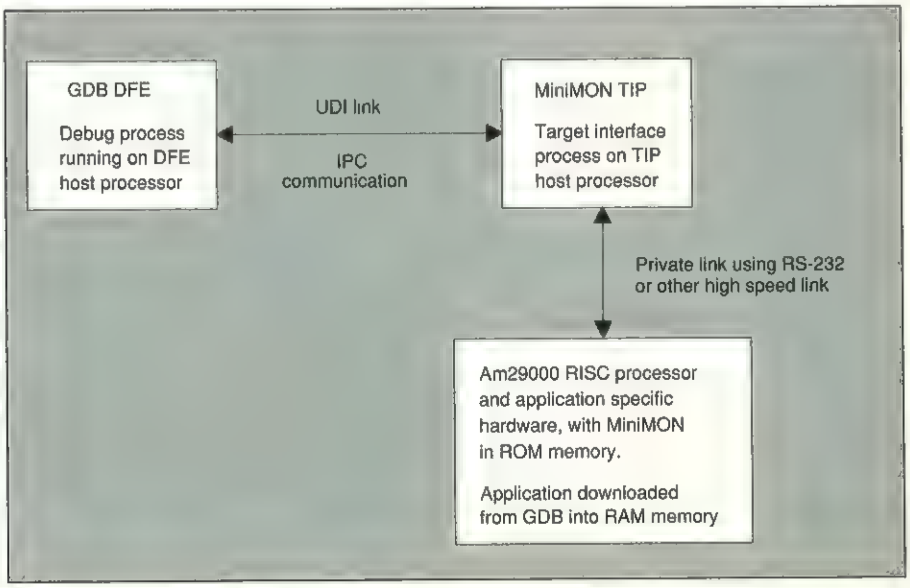


Figure 3 -Architecture of MiniMON

function call to the TIP process. The TIP code developer must resolve the function request, by adding code which is specific to controlling the particular target. The IPC layer is effectively transparent, the TIP developer is unaware that the procedure caller is from a different process, possibly on a different host machine. Figure 4 lists most of the UDI-p procedures available.

Because the DFE and TIP processes may be running on different machines, care must be taken when moving data objects between hosts. An `int` sized object on the DFE supporting machine may be a different size from an `int` on the TIP supporting machine. Further, the machines may be of different endian. The UDI-p procedures make use of a machine independent data description technique similar to the XDR library available with UNIX. Data is converted into a universal data representation (UDR) format before being transferred via sockets. On being received, the data is converted from UDR format into data structures which are appropriate for the receiving machine. The UDI-p procedures keep the UDR activity hidden from the UDI user.

I will now discuss, in more detail, the development of a UDI conforming GDB, a source level debugger from the Free Software Foundation. GDB is an example of a DFE process. As an example of a TIP process, we shall look at the MiniMON monitor for the Am29000 RISC processor. Most users of GDB will have some knowledge of the `ptrace()` system call which enables GDB to examine the state of the process being debugged. A brief description of `ptrace()` is useful, along with further explanation of its unsuitability for embedded application software development.

GDB to UDI

`ptrace()` is a UNIX system call which provides a means by which a process may control the execution of another process executing on the same processor. The process being debugged is said to be *traced*. However, this does not mean that the execution path of a process is recorded in a *trace buffer* as is the case with many processor emulators. Debugging with `ptrace()` relies on the use of instruction breakpoints and other hardware or processor generated signals causing execution to stop.

```
ptrace(request,pid,addr,data)
```

The interpretation of the arguments depends on the `request` argument. Generally, `pid` is the process ID of the traced process. A process being debugged behaves normally until it encounters some signal, whether internally (processor) generated like an illegal instruction exception, or externally generated like an interrupt. The traced process then enters a stopped state and the tracing process is notified using the `wait()` system call.

The goal of UDI is to provide a standard interface between the debugger developer and the target communications module

When the traced process is in the stopped state, it's core image can be examined and modified using the `ptrace()` service. If desired, another `ptrace()` request can then cause the traced process either to terminate or to continue. Figure 5 outlines the `ptrace()` request services available.

In an embedded system, the user-interface process, which controls the debugging, and the application process, which is being debugged, may not be executing on the same processor. So it is not possible to use the `ptrace` system call mechanism to debug embedded application software. The debugger process (DFE) must run on a separate processor and communicate with the processor supporting execution of the application code.

The Free Software Foundation's source level debugger, GDB, by default makes use of the `ptrace` system call. However, it can alternatively use a collection of procedures which support communication to a remote processor. These procedures implement the necessary protocols to control the hardware connecting the remote processor to the host debug processor. By this means, GDB can be used to debug embedded application software running on application-specific hardware.

Newer versions of GDB (version 3.98 and on) implement the procedural interface to a remote target processor via procedure pointers which are members of a `target_ops` structure. The procedures currently available are listed in Figure 6. According to GDB configuration convention, the file *remote-udi.c* must be used to implement the remote interface procedures. In the case of interfacing to the IPC mechanism used by UDI, the procedures in Figure 5 are mapped into the UDI-p procedures given in Figure 4. With the availability of the UDI-p library it is a simple task to map the GDB remote interface procedures for socket communication with a remote target processor.

UDI to MiniMON

AMD's MiniMON is an example of the kind of software required at the business end of the setup. MiniMON is not intended to be a stand-alone monitor - it requires the support of a software module - the target interface process (TIP) - located in a support processor. The Am29000 target processor communicates with the processor running

Procedure	Operation
UDIConnect	Connect to selected TIP
UDIDisconnect	Disconnect from TIP
UDISetCurrentConnection	For multiple TIP selection
UDICapabilities	Obtain DFE and TIP capability info.
UDIEnumerateTIPs	List Multiple TIPs available
UDICreateProcess	Load a program for debugging
UDISetCurrentProcess	Select from multiple loaded programs
UDIDestroyProcess	Discontinue program debugging
UDIInitializeProcess	Prepare run-time environment
UDIRead	Read data to target processor memory
UDIWrite	Write data to target processor memory
UDICopy	Duplicate a block of data in target memory.
UDIExecute	Start/continue target processor execution
UDIStep	Execute the next instruction
UDIStop	Request the target to stop execution
UDIWait	Inquire about target status.
UDISetBreakpoint	Insert a breakpoint
UDIQueryBreakpoint	Inquire about breakpoint
UDIClearBreakpoint	Remove breakpoint

Figure 4 - Principle UDI-p procedures

the TIP process via a serial link or other higher performance channel. This link supports a message system which is 'private' to the MiniMON monitor, by which I mean that it is completely independent of the UDI protocol. See Figure 3.

Developers of software for embedded applications are used to working with emulators. They enable code to be downloaded to application memory or installed in substitute overlay memory. This avoids having the development delays associated with running code from EPROM. Emulators may be indispensable in the early stages of getting the target hardware functional. However, once the processor is able to execute out of target system memory and a communications channel such as a serial link is available, the need for an emulator is reduced. Emulators are expensive, and it is not always possible to make one available to each team member. The use of a debug monitor such as MiniMON during the software debug stage of a project is an economical alternative to an emulator.

MiniMON must be installed in target system ROM memory or downloaded by the host via a shared memory interface. The target application code and additional operating system code can then be downloaded via the message system. If changes to the code are required, then the message system can be used to download new code quickly without changing any ROM devices.

Most monitors do not offer high-level language support. Debugging takes place at the level of assembly code instructions rather than the original, say C, code. Using GDB in conjunction with MiniMON enables source level code to be debugged, which is far more productive and necessary for large software projects.

Summary

A number of debug tool developers are currently or will be shortly offering tools which are UDI compliant. Typically the DFEs are C source level debuggers. This is not surprising, as the increased use of RISC processor designs has resulted in a corresponding increase in software complexity. The use of a high-level language such as C is more productive than developing code at machine instruction level. And further, the use of C enables much greater portability of code among current and future projects. The low cost of GDB makes it an attractive choice for developers.

Target processors and their control mechanisms are much more varied than DFEs. I have briefly described the MiniMON TIP, which is a process which controls the execution of an Am29000 processor. A small amount of code known as the *debugcore* is placed in target processor ROM memory and enables examination of the processor state. The MiniMON TIP communicates with the debugcore via a hardware link which is specific to the embedded application hardware.

Other TIPs already exist and are under development. I know of an Am29000 simulator (ISS) which runs on UNIX hosts. The DFE communicating with the simulator TIP is unaware that the Am29000 processor is not present, but being simulated by a process, executing on, say, a UNIX workstation. There are also tool developers constructing TIP programs to control processor emulators. This will make possible a top-of-the-line debug environment.

Because debuggers like GDB are available in source form, developers can add additional debug commands, such as examination of real-time operating system (OS) performance. This would require adding OS structural information into GDB. When the debugger front end and, for example, emulator interface module are supplied as a single executable, adding new commands is not possible. Via the use of Internet sockets the debugger may execute on a different networked host than the node supporting the emulator control process.

UDI makes possible a wider tool choice for application code developers. Debugger front end tools are supplied separately from target control programs. The user can consider cost, availability and functionality when selecting the debug environment. Never before has this level of debug tool configurability been available to the embedded application development community.

EXE

Request	Operation
TraceMe	Declare that the process is to be traced
PeekText	Read one word in process's instruction space
PeekData	Read one word in process's data space
PeekUser	Examine the process control data structure
PokeText	Write one word in process's instruction space
PokeData	Write one word in process's data space
PokeUser	Write to the process control data structure
Cont	Start-up process execution
Kill	Terminate the process being debugged
SingleStep	Execute the next instruction
GetRegs	Read processor registers
SetRegs	Write processor registers
ReadText	Read data from process's instruction space
ReadData	Read data from process's data space
WriteText	Write data into process's instruction space
WriteData	Write data into process's data space
SysCall	Continue execution until a system call

Figure 5 - The ptrace( ) request services

Function	Operation
to_open()	Open communication connection to remote target
to_close()	Close connection to remote target
to_attach()	Attach to a loaded and running program
to_dettach()	Detach, for multi-target debugging
to_start()	Load program into target system memory
to_wait()	Wait until target system execution stops
to_resume()	Start-up/continue target system execution
to_fetch_register()	Read target system processor register(s)
to_store_register()	Write register(s) in target system processor
to_xfer_memory()	Read/Write data to target system memory
to_insert_breakpoint()	Establish an instruction break address
to_remove_breakpoint()	Remove a breakpoint
to_load()	Load a program into the target processor memory

Figure 6 - GDB Remote Target-Operations

Daniel Mann is senior member of the technical staff supporting the Am29000 processor at AMD in Austin, Texas. He bears an unexpected British accent, and may be contacted on 0101 512 462 4872, or via Email as `daniel.mann@amd.com.`

The XRAY Debugger Front End is produced by Microtech Research (0256 57551), the CDB Debugger Front End is produced by Third Eye Software.

Configuration Management

PVCS and PolyMake for UNIX platforms...

- DEC
- Bull
- DataGeneral
- ICL
- Motorola
- NCR
- Nixdorf
- Prime
- Siemens
- Unisys
- and others...

Readmar Systems
L I M I T E D

Tel (+44) 071 625 5255
Fax (+44) 071 624 9404

CIRCLE NO. 578

THE INSTITUTION OF ANALYSTS & PROGRAMMERS

The Institution of Analysts & Programmers represents an elite body of men and women who are leaders of the computing profession. These are people whose expertise enables them to analyse the problems of modern industry, and apply computers to their solution.

Membership of the Institution is a recognised mark of professional status. Designatory letters, which members are entitled to use, indicate their grade within the Institution, and their standing within the profession. Grading depends on age, experience and academic attainment.

Applications are welcomed from all men and women who are engaged in systems analysis or computer programming, or who are training for the profession. Enquiries may be made by letter, telephone or fax.

Telephone 081-567 2118 ★ Fax and Messages 081-567 4379

The Institution of Analysts & Programmers
Charles House, 36 Culmington Road,
London W13 9NH. England

CIRCLE NO. 545

"Enough to make you a LATE DEVELOPER"... PC Business World July 91

Imagine a powerful integrated development environment common to whichever languages you choose, capable of supporting DOS, Windows or OS/2 development embodying the latest in OOP technology – it's called ***TopSpeed*** from JPI. Its modular architecture opens up new realms of choice – simply slot-in languages, source libraries, and toolkits to build a development system that meets your needs. No redundant components – what you want is what you get!

***TopSpeed* Environment** – Multi-window editor, powerful Project system, Hypertext help (environment, all languages and library), debugger, syntax checker, menu or command line driven, EMS support, pop-up calc, plus . . . plus . . . DOS or OS/2 **£59**

***TopSpeed* TechKit** – For power programming. Supports DOS DLLs, post mortem debugging, advanced overlay manager, TopSpeed assembler, .EXE file compressor, .OBJ file disassembler. **Windows 3 resource compiler**, Windows 3 run-time libraries, plus . . . plus . . . **£59**

***TopSpeed* C++** – makes C++ lean and mean Unique to JPI – TopSpeed is the only true AT&T 2.1 C++. TopSpeed C++ gives you SmartMethod® Linking which eliminates unreferenced classes, methods and even virtual methods – a real breakthrough for OOP programming. Includes short based pointers, concurrent tasking – even with DOS, plus . . . plus . . . DOS or OS/2 **£59**

***TopSpeed* C** – *the standard is enhanced.* The only ANSI certified C. Generates compact high quality code for DOS, OS/2 and Windows 3. Includes run-time error checking, multi-threading, mixed memory models, links to C++, Modula-2 and Pascal, plus . . . DOS or OS/2 **£59**

***TopSpeed* Modula-2** – the world leading implementation of Modula-2. This strongly typed and highly structured language includes type safe conversion between objects, OOP extensions with true multiple inheritance, virtual pointers, based pointers, links to C, C++ and Pascal, plus . . . DOS or OS/2 **£59**

***TopSpeed* Pascal** – the next generation. Power-up your Pascal, convert to TopSpeed, then tune it up with the hottest compiler for DOS, Extended Dos, OS/2, or Windows. ISO 7185 conformant, Turbo to TopSpeed converter, ISO conformant arrays, dynamic strings, separate compilation units, OOP extensions plus much more . . . DOS or OS/2 **£59**

Library Source Kits available in all languages

Call JPI on (0234) 267500 now for your free copy of the TopSpeed 1992 Compiler Catalogue

TopSpeed products are available from:
GreyMatter 0364-53499 · System Science 071-833-1022
RTA 081-656-7333

JPI
3 The Mansards, Tavistock Street, Bedford MK40 2RX
Fax: (0234) 217094

CIRCLE NO. 546

TopSpeed® C++ TopSpeed® Modula-2 TopSpeed® Pascal TopSpeed® C

ix

programming editor for Windows

◆ Full Windows implementation

Cut & Paste, nearly infinite Undo, Point and Click or key strokes control all features

◆ Multiple files and windows using MDI

Load and view several files simultaneously in separate windows

◆ Folding text

Manage large object modules with sections of text 'folded' up into a single line

◆ Code documentation

Add 'post-it' style notes to any file

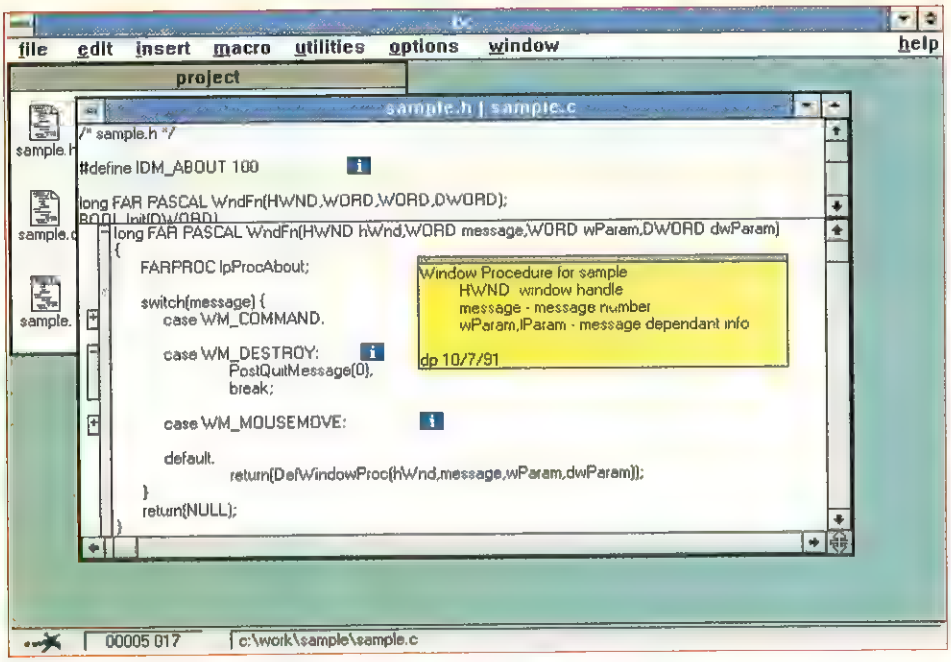
◆ Fully reconfigurable keyboard and menus

Re-assign all keys and menus, make multiple key assignments

◆ Powerful macro language built-in

Create and compile macros using any editor function with a Basic-like language

◆ £95 + VAT



As more and more users switch to Windows 3, the demand for Windows software continues to grow. Yet much of the developer's toolkit still runs under DOS. Writing for Windows has always been more complex than for DOS, and project management can be especially tricky as Windows programs tend to be made up of large numbers of files.

ix was developed to enhance programmer productivity. ix runs under Windows, and uses all the standard GUI features – MDI, split windows, cut & paste, choice of fonts and colours. It also contains several unique features which allow the developer far greater control.

With ix you can edit and test within Windows – no more switching in and out of a DOS editor. Compilation/make files can be launched from a 'Browse' window.

Browse windows help you keep track of the multiple files of a Windows project. A browser can be created for each project and used to fire up edit windows, launch resource editors (such as SDKpaint) and run programs as in Program Manager.

'Post-it' style notes can be stuck onto any source file – without modifying the file. Their names can be used to give a hypertext-like help system throughout your project.

And there's more; our AdHoc macro language, reassignable keys, custom menus and free telephone support, but we haven't got space to tell you about all that here. For more information, call or write us and we'll send you our information pack.

positive

• positive limited •
• 22 westminster buildings •
• new york street •
• leeds ls2 7dt •
• 0532 343 104 •

CIRCLE NO. 547

A UNIX fit to embed

UNIX is too fat, slow and complex for use in real time applications. David Hann explains how to trim excess fat.

UNIX is now established as the Open Systems operating platform and the move towards using UNIX has begun, or is complete, in many IT industry sectors. There is, however, one group of application developers who may well feel left out in the cold with respect to the ground swell of UNIX fever. For real time control and data acquisition applications, typical of the telecommunications, robotics and simulation industries, UNIX has no help to offer. Although the benefits of UNIX are just as appealing, there are two main problems that prevent its deployment:

- Real Time Behaviour - UNIX was designed as an interactive operating system, giving a good response to a large number of logged-in users. UNIX lacks the predictable real time response needed to safely control plant or to acquire data at high speeds.

- Stand-alone Operation - UNIX is notorious for the resources it consumes, both in terms of simple hardware (memory, disks, backup devices etc) and in terms of system administration overhead. Both of these factors make UNIX unsuitable for remote or stand-alone operation.

This article discusses the approach taken by VenturCom Inc in producing VENIX/386, a real time PC UNIX with an embedded capability for stand-alone operation.

What are they?

The term 'embedded system' encompasses a wide range of applications. Most embedded computer systems are involved in the monitoring and controlling of real world processes and systems in a full-time, dedicated fashion. Embedded systems are often required to interact with human operators and with other, higher-level information systems. A system could equally be totally unattended and in some remote location or harsh environment.

At one end of the embedded system scale, there are relatively simple applications that contain less than 100 KB of software. Clearly UNIX (or any other fully-functional OS) is inappropriate for these situations. The application simply does not justify the overhead.

At the other extreme, complex applications with sizes of more than several hundred kilobytes require full-featured operating systems. These involve advanced networking interfaces and connections, and perhaps graphical user interfaces. They use off-the-shelf third party software such as database management systems, and employ distributed control and monitoring. These types of applications continually adapt in response to evolving needs.

To many users already familiar with UNIX, the idea of shoe-horning it for use as an embedded application environment seems as absurd as it is it does impossible. UNIX systems typically require in excess of 40 MB of disk space before loading any applications, and a minimum of 4 MB of memory without X windows. For many stand-alone real time systems, this represents an unacceptable hardware cost. Furthermore, system administrators, familiar with the nuts and bolts of UNIX, must be permanently on-hand and the operating system is extremely vulnerable during power loss or hardware failure.

Despite these undoubted problems, UNIX offers a great number of benefits. These include standards, networking, user interface systems, and access to third-party off-the-shelf applications. Perhaps the biggest potential benefit is the ability to maintain a homogeneous operating environment with respect to other larger systems. Hence the primary goal for embedded UNIX, and the real reason to use UNIX in the first place, is to maintain the same basic application environment as standard UNIX. Thus developers benefit from previous UNIX expertise, use of third party and previously developed software, and a common, powerful development and execution environments. After the application design is understood and specified, then a compact target system is configured which includes only the operating system software and utilities used by the application.

In addition to slimming down the target system, UNIX must be enhanced for acquisition and control, a robust and unattended operation. These enhancements can be done without significantly changing the application environment, except for the obvious issues such as unattended applications which cannot expect a local console for operator dialogue. Much of the following discussion is based upon the development of VenturCom's ROM and embedded VENIX/386 products.

The surgeon's knife

Even though complex embedded applications require more software and hence more memory, any superfluous memory usage drives up cost and system complexity. Most UNIX applications depend on only a handful of utilities, and many embedded applications require no utilities. Unused utilities and software are discarded, although indirect dependencies (utilities which depend upon other utilities) can complicate matters. In the best cases, this first pruning step results in a 2 MB to 4 MB file system. This residual size includes the kernel and system administration of login utilities.

Since embedded systems start application execution immediately after power-up, there is no need for login software. System administration on embedded systems is less comprehensive and simpler than on full UNIX systems. Also, standard UNIX administrative utilities are inappropriate for operators and service personnel of an embedded system, and are not designed for

unattended use. Therefore we must remove all of the standard administrative utilities, except a few for certain configurations (and, of course, those utilities called by the application). All the removed functionality is replaced by a new '/etc/init' utility, which is very compact and robust. The residual size requirement for the file system is now well below 500 KB, including the kernel and the /etc/init files.

Over 1 MB of RAM is required for the standard UNIX kernel code and data structures, such as buffer cache and page tables. By removing unused device drivers and file system types, reducing the number of buffers, and replacing a bloated console driver with a smaller cousin, we can halve the memory requirements. In some cases, applications do not use streams or require paging. By making the kernel more modular and configurable, total kernel sizes of below 300 KB are routinely achieved for a UNIX System V, R3.2 running on an 80386 machine.

Finally, tools are required to assist developers with all these configuration steps to reduce file system and RAM sizes. Based on target computer hardware configuration and a dialogue with the application developer, these tools configure and generate the kernel, include relevant utilities along with application software, and create a bootable image. Possible media for the image include floppy diskette, winchester disk, ROM devices, and network booting from a host computer.

Determinism

Monitoring and control of real world systems requires determinism. If the computer controlling a robot arm does not sense and update the arm trajectory at precisely required times, then the robot may well mangle the product it is supposed to be assembling. Although the level of determinism required varies between applications, in general the faster it is the better. Standard UNIX has unbounded worst case response times, but may be acceptable for slower applications with response times of a second or longer. UNIX with real time extensions is required for most applications.

The key factor in determinism of UNIX systems is kernel preëmption. Preëmption is the ability to suspend one process when an event (such as a device interrupt) occurs, and context switch to execute a higher priority process. Standard UNIX does not preëmpt while a process is executing a system call, unless the system call blocks 'waiting for I/O completion'. Since some system calls require extensive, and essentially unbounded kernel-level processing, determinism is unbounded. Measurements show frequent cases of over 100 millisecond response times, and occasional times of over 1 second.

UNIX is notorious for the resources it consumes, both in terms of simple hardware and system administration

Adding checkpoints in the kernel (places where the kernel checks for higher priority processes) effectively bounds the worst case response time. Introducing 20-30 checkpoints at strategic locations in the kernel will produce worst case response times of 2-5 ms. Making the kernel fully preëmptable, that is preëmpting immediately after the event, lowers the worst case response time to 200-500 µs.

Unattended Operation

After power-on or system reset, the kernel must boot and the environment must be configured and set up before the application can begin execution. Activities include determining the hardware configuration such as the presence of networks and other devices, checking switch settings, mounting data storage devices, and initialising network services. Depending on the success or failure of these activities, different steps and applications may be required. The replacement '/etc/init' utility mentioned earlier is designed to handle all of these exercises from a self-contained script, without any operator intervention.

ROM

Embedded targets of particular interest are ROM-based systems. They often represent the extreme of dedicated and robust (no disk or tape drives to fail) systems, and place serious constraints on application and operating system software. Previously, large applications or applications subject to frequent change were never placed in ROM. Today, high capacity, field reprogrammable, low cost ROM makes UNIX based applications in ROM a viable option to system developers.

The ROM of target computers is divided into two regions. The first is a small ROM (often high speed) which contains the self-test and start-up code that is executed when the computer is powered up (included in the BIOS of PCs). This start-up code loads the UNIX kernel from the second ROM region. Kernel or application code is not run directly out of this ROM because of the often slower ROM memory speeds, and because some ROM devices do not present a uniform physical address apace (windowed ROM).

The kernel, which is modified to work with a read-only root file system and without paging, accesses the second ROM region as a read-only disk device containing a normal UNIX file system. In fact, the kernel is simply a file in the ROM file system. Many existing single board computers already support a bootstrap which loads an operating system from a ROM as if it was a disk.

Preparing a ROM target uses the same embedding tools and steps as embedding UNIX for disk operation (except for the use of a slightly different kernel and the additional step of 'burning' the ROM). A minimal system configured for ROM operation consists of an 80386 processor, 1 MB of RAM, 1 MB of ROM, a network connection and an interface to an instrument or device.

Conclusion

Under the guise of Open Systems, UNIX has evolved out of its initial niche in the software engineering community into the wider computing arena. Real time capabilities have taken UNIX to scientific and laboratory users, and then to the broader engineering acquisition and control markets. Now the next step of embedding UNIX systems is taking place. The feasibility of embedded UNIX has been demonstrated with the successes of the early adopters. Despite the fact that the use of embedded UNIX is still in its early stages, technical improvements in tools for embedding, system robustness and fault-tolerance, and real time, will continue to open more applications to the benefits of UNIX. The trend is clear. As embedded applications become more complex and open systems more prevalent, embedded UNIX will become the solution of choice.

EXE

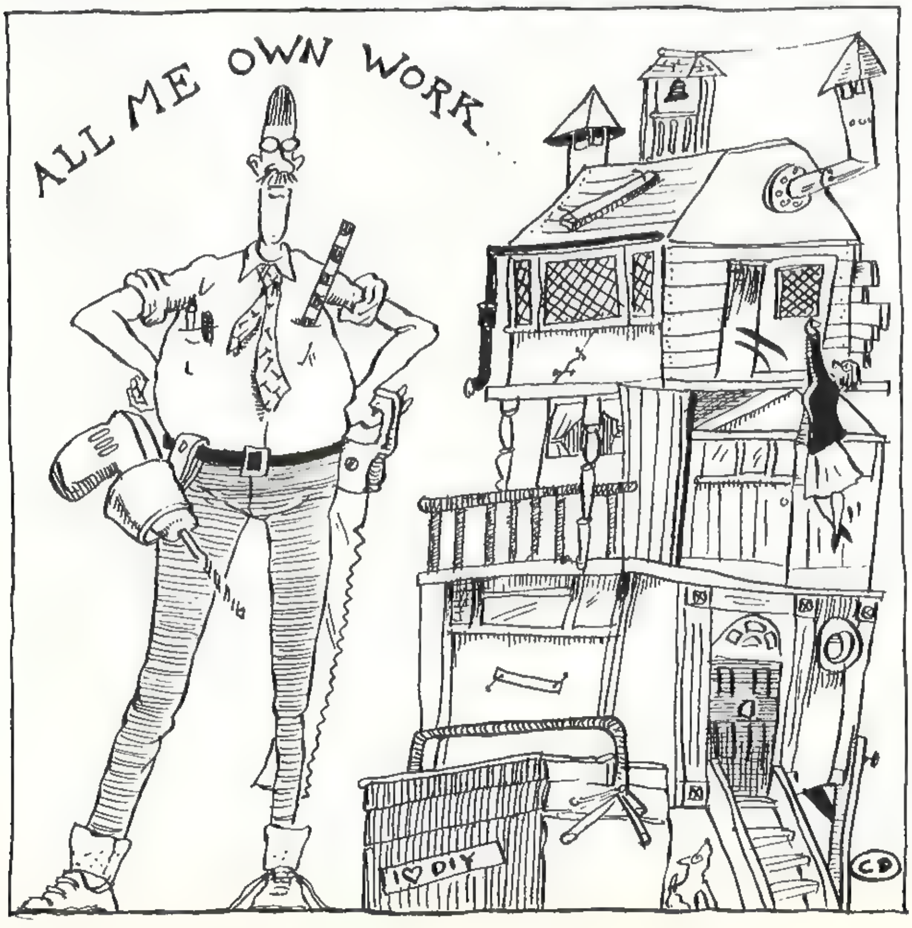
David Hann is Managing Director of Real-Time Products, which distributes VENIX in the UK. He may be contacted on 021 333 6955.

Don't DIY

Designers of embedded applications frequently use an in-house operating system. Microware's Stephen Montgomery makes the case for buying-in an OS.

The fundamental task of an operating system is to supervise the resources and functions of a computer. This includes providing an interface between the computer and the outside world (user), managing the system's I/O, creating and managing a file system for data storage and retrieval, managing both the system and user memory and providing a method for executing programs. As such, the term can cover a whole range of firmware for computers ranging from a small microcontroller to the largest mainframe, and so from the simplest in-house scheduler to the most sophisticated supervisory system. The most common operating systems are found on general purpose PCs and minicomputers and are very well known: MS-DOS, UNIX, VMS, MVS etc. But systems suitable for embedded processor applications on single board computers or small subsystems are available and provide a powerful and comprehensive range of benefits to the user.

There are many of these systems available, each with its own characteristics and suitability for a particular application. Such names as C EXECUTIVE, Flexos, LynxOS, MIRAGE, PDOS, pSOS, QNX, RMX, RTOS-UH, VENIX, VMEexec, VRTX and VxWorks, not to mention Microware's own OS-9 and OS-9000, can provide attractive propositions for the designers of ever more complex and sophisticated applications, especially where real time operation is desired with multi-tasking capability.



From the earliest days, designers of embedded systems of microprocessor engineering have traditionally written their own scheduler to manage the single or multiple tasks on a processor. In many cases application complexity has increased and the scheduler has grown in complexity alongside it. Usually these in-house schedulers are written from scratch for each individual project within a company, tailored specifically for the job in hand and not written for universal application. This results in a range of different schedulers which, even with the best intentions, are unlikely to be documented extensively enough for easy system maintenance to be carried out.

The operating system, like the processor hardware itself, is something that can be bought 'off the shelf' as a standard product, allowing the development team to concentrate on the specific application rather than getting entangled in the basic system - multitasking, I/O management, inter-process communication, memory management. As with any proposition there are alternatives and the decision to develop or buy has to be made on a range of criteria. In this article, I hope to make you see that there are very few valid reasons why an in-house operating system is preferable to a bought-in one - at virtually any level of system complexity, performance or production level.

Suitability

An in-house scheduler will be characterised for the system it is designed for and will contain the features required by the system. A general purpose operating system can be selected which will cover these requirements, plus a lot more which can just be ignored (at the expense of some ROM space). However, as the application is developed and further capability added, unused features can be utilised. With an in-house system written for a specific project and defined at the very start, it is possible that the addition of enhancements may not be possible,

simply because the scheduler just cannot accommodate an extra task or feature without a major rewrite.

Speed and size

One of the first questions asked of proprietary operating system vendors is: how fast is it? The implication is that the designer could do better in his dedicated system and speed is a critical factor. I concede that speed is one of the few areas where an in-house scheduler can do better than a bought-in rival. But these days there are a choice of operating systems available, from a bare minimum to a full operating system, and a sufficiently fast one exists for all but the most demanding applications.

Ironically, speed is also one of the few areas where system specification is a problem and a potential stumbling block to the system designer. Benchmarks are not standardised, and some vendors have exploited this by offering 'bare-bones' kernels that are optimised to perform simple tasks quickly. While these kernels are fast and predictable they don't actually do much. It is worth noting that a kernel is just that, and speed is sacrificed at the expense of capability. So when comparing interrupt latency times, for example, remember that additional code will be necessary as an overhead of the handler that services that interrupt.

That having been said, kernels may offer the best solution for a particular requirement and should not be overlooked. Remember that system designers using kernels may be forced to create and embed much of the operating system functionality in their application.

As for size - obviously a scheduler written for the particular application will only contain relevant code and should require less ROM space than the smallest suitable general purpose kernel or operating system. But is this a real issue? With the level of memory density now available, and given that the kernel of an operating system such as OS-9 requires only 256 KB, this is not usually a problem. Because they are continuously being developed and optimised, commercial operating systems are very code efficient. Superior design, for example using re-entrant modules so that only single copies of each type of code module is required, where a more hastily developed in-house scheduler might require multiple copies. What you lose from lack of tailoring to your particular circumstances, you may well gain from the use of well-honed code.

Growth

The ability to provide functions over and above those needed to meet the original specification is not always obvious at project initialisation. Most (if not all) system designers have surely been asked to add extra features to the product once they have started. If it is just a slight enhancement which can be carried out by modifying application code, the cost just minor aggravation (and a chance to moan about the marketing department!). But if it involves adding an additional task or I/O type it can be a major headache, bordering on the impossible. This is especially the case if the request comes years after the original design team has been disbanded or critical members left the company.

Proprietary operating systems have spare capability to accommodate these requests and their general purpose nature makes the addition of extra tasks a trivial matter. Even adding new features - a disk to a diskless system, networking capability etc - can often be as simple as adding the appropriate file manager, standard device driver and a code module to manipulate the data for communication between processes. And because the operating systems are fully documented, the engineers following on have some chance of understanding the system.

Manufacturers of these systems are continually adding enhancements. If a new type of device becomes available - SCSI disks and NFS are recent examples - the manufacturer can usually supply appropriate drivers and revised versions of code so that the system maintainer can easily and quickly be added to the system.

Costs

At first sight a bought-in operating system may seem expensive, especially on large production runs. But so too are programmers and designers. An in-house scheduler can easily absorb a man-year of effort to design, build and document, and that doesn't include additional maintenance. Current engineering costs are in the region of £35,000 per man-year - these can easily double if contractors are employed. That amount of money can buy a lot of copies of a kernel or operating system. A ROMmed system containing, for example Industrial OS-9 for the 68000 chip (consisting of kernel, character I/O support and file handler) costs approximately £70 per copy at 50 off quantities, reducing to £25 at 1000 off orders.

The point here is that commercial operating systems should not be ruled out on cost grounds alone without a very thorough and critical analysis of the costs and other benefits associated with them when compared with an in-house design.

Reliability

The maxim that no software is bug-free leads to an advantage in proprietary operating systems, which should contain well tried and tested code. Clearly no vendor can guarantee his system to be entirely bug-free. However, if several hundred thousand copies are in everyday use and the system

has undergone continual development throughout its existence, it is safe to assume that bugs that do remain lurking deep in the code are unlikely to be encountered and so just as unlikely to cause catastrophic failure. The reliability of an in-house scheduler, especially a real time multi-tasking one, where it is impossible to extensively test for the whole range of extraneous conditions incident on the system, is far more questionable and might prove fatal.

Development

The ability to test an application program running under an operating system with tools optimised for that system is extremely attractive in terms of overall system proving and confidence. No matter how simple an in-house scheduler is, it is unlikely that a comparable range of tools exists. This in itself can save considerable development time. Some vendors supply cross development tools based on UNIX and PC hosts which allow strategic software development procedures and accounting packages to be employed - even for the lowest level embedded system.

How to cope with the forward rush of hardware technology? The DIY man faces the prospect of porting both his application and his scheduler to any new platform. On the other hand, a commercial operating system is designed to be targeted on a variety of processors in a single form. Because manufacturers generally commit their products to new processors as soon as they become available, it is simple to take an application through a range of processors of present and future designs. For example software for the 68000 could be switched to the 68020 then the 68030 as greater processing power is required and the chip prices drop. People are doing this regularly. As the 68040 becomes available, the same code will run on it at the drop of a recompilation. Some operating systems can even compile code across different manufacturers' products, so it is theoretically possible to delay until after the software is written not only the choice of generic processor type (eg 68030/68040) but also the manufacturer and type of processor (eg Intel/Motorola).

Conclusion

While the initial cost of buying an operating system may look high in terms of monetary investment, training and development tool commitment, it is unlikely to be more expensive than designing one's own scheduler. And when looked at as a whole, the range of benefits that accompany the bought-in kernel or operating system far outweigh the in-house scheduler over both short and long terms.

In essence the operating system should be considered as hardware: look for the most suitable, invest in training and tools, standardise throughout the company and reap the benefits of quantity, flexibility and understanding. That way expensive engineering effort can be concentrated on the specific application to yield a better product with a faster time-to-market.

EXE

Stephen Montgomery is Technical Sales Manager at Microware UK, and is a member of the IEE. He can be contacted on 0703 601991.

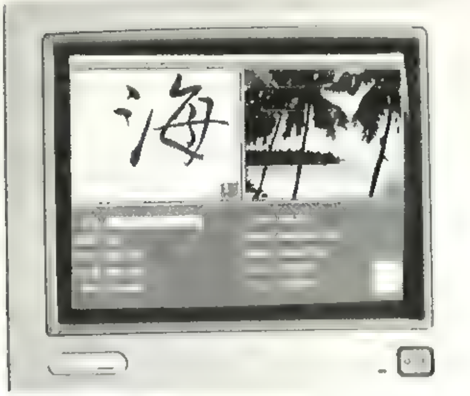
In the interests of fair play, the Editors of .EXE would be most interested to hear from anybody whose experience is at variance with the situation as described by Mr Montgomery.

Escape with C-scape

ELEGANT GRAPHICS AND TEXT

C-scape Features

- Graphics — Combine high-resolution graphics with text or menus.
- Object-oriented — Add features and create re-usable code modules.
- Mouse — Use any standard mouse for fast screen control.
- Portability — Write hardware independent code. Supports MSDOS, OS/2, UNIX and VMS. Auto-detects: Hercules, CGA, EGA, VGA.
- Text Editing — Create a full-featured text editor or pop-up note pad.
- Field flexibility — Create masked, protected and marked fields with complete data validation. Use time, date, money, pop-up list and many more functions — or create your own.
- Windows — Choose from pop-up, tiled, bordered and exploding windows with size and numbers limited only by RAM.
- Menus — Choose from pop-up, pull-down, 123-style or slug menus — or create your own.
- Context sensitive help — Link help messages to individual screens or fields. Cross reference messages to create hypertext-like help.
- Screen design — Build any type of screen or form with Look and Feel™ screen designer, which will then automatically convert to C.
- Screen flexibility — Call screens from files at run-time or link them.



The **C-scape**® Interface Management System frees C programmers from the tedium of coding windows, menus, data validation, help and text editing functions.

C-scape is a pleasure to use. With **C-scape**'s object oriented design you can build more flexible, more functional, more portable and unique applications — and you will enjoy doing it.

The Industry Standard.
Many thousands of programmers have given up home-grown libraries and cumbersome, inflexible products for the ease of **C-scape**. In the US, *IEEE Computer* said "**C-scape** is by far the best . . . a joy to use." *PC Magazine* chose **C-scape** to produce its Laboratory Benchmark Series 5.0 software because **C-scape** offers mouse support. Moreover, **C-scape** simultaneously combines text and graphics.

C-scape, from LIANT SOFTWARE CORPORATION, is built around an open architecture, so it can be used with data management or other C libraries. At Systemstar, we offer **C-scape** with **Raima Data Manager** from Raima Corporation to provide a complete development environment.

To port your application from MSDOS or OS/2 to UNIX or VMS, just recompile.

Source code is included in the price and there are NO RUN-TIME ROYALTIES.

For more information about C-scape call Systemstar on (0992) 500919.

SYSTEMSTAR LIMITED
1-3 Parliament Square, Hertford, SG14 1EX
Telephone: (0992) 500919 Facsimile: (0992) 554261

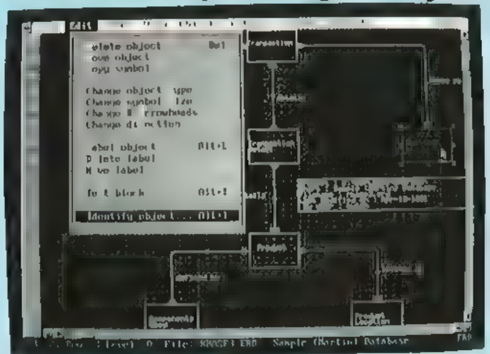
CIRCLE NO. 555

EasyCASE™ Plus

NEW! VERSION 3.0 Call for upgrade details.

The affordable approach to software engineering... only £320*

Finally, there's a CASE tool that won't get in the way of your creativity... A tool that makes structured analysis, structured design and data modeling as easy as working with any other tool on your PC - EasyCASE Plus! Using EasyCASE Plus' new, easy to use graphical user interface (GUI), you'll be creating and editing charts, linking them, and building your data dictionary in no time. As well as being easy to use and easy to learn, EasyCASE Plus is easy on your budget! Ask any user. They'll tell you it's the best buy for your PC based CASE tool needs. Discover why over 4,000 software professionals use EasyCASE Plus and how you can join them!



"EasyCASE Plus is a well designed, low priced tool that is easy to learn and provides excellent diagramming capabilities... EasyCASE Plus is an excellent investment."

COMPUTER LANGUAGE PRODUCTIVITY AWARD 1990

Features:
- IBM SAA/CUA compliant graphical user interface (GUI)
- Extensive diagram editing features
- Integrated dBASE III compatible data dictionary
- Integrated dictionary manager, reports manager, process editor
- Hierarchical chart linking & process decomposition
- Record and element definitions
- Extensive printer, plotter and desktop publishing support
- Data dictionary import, export, and merge
- On-line help
- Comprehensive documentation with tutorial
- Access to your database, word processor, DOS, etc.
- Integrated diagram analysis (optional)

Requirements:
Runs on: IBM PC or PS/2 (AT recommended), DOS 3.1 or higher, EGA/VGA color, mouse, 640 K RAM (500 K free), 1 MB EMS recommended, math co-processor supported. Printers/Plotters Supported: Epson FX & LQ, IBM Graphics & Proprinter X24, HP QuietJet, DeskJet, & LaserJet, HP Plotters, PostScript.

EasyCASE Professional £420*
(includes integrated DFD level balancing and data dictionary/diagram analysis)

Methods:
- Yourdon/DeMarco
- Gane & Sarson
- SSADM (DFDs)
- Ward-Mellor/Hatley
- Yourdon/Constantine
- Martin
- Chen, Bachman

Diagram Types:
- Data Flow Diagrams (DFDs)
- Structure Charts
- State Transition Diagrams
- Entity Relationship (ERDs)
- Data Model Diagrams
- Transformation Schema (real-time DFDs)

Evergreen CASE Tools

THE SOFTWARE CONSTRUCTION CO. LTD.
FAX: (0763) 244025

Call today for a brochure!
Tel: (0763) 244114

1 The Maltings, Green Drift, Royston, Hertfordshire SG8 5DB

CIRCLE NO. 604

Yet More Bjarne

Will Watts ambushed C++ inventor Dr Stroustrup with a recording Walkman during a recent visit to London to speak to the European C++ Users Group. Here is what he had to say.

What is the situation with the ANSI and ISO C++ standardisation committees?

Given the experience that other languages have had with standardisation, we had every reason to fear trouble. Since at least the days of FORTRAN, there seem to have been institutionalised fights going on between the various national bodies. But we had the first ISO meeting in Lund last summer, and we agreed to hold joint meetings and produce one document.

I have high hopes that standardisation will go smoothly. The process worked in Lund, it worked in Dallas a few weeks ago and we just expect it to go on.

When can we expect some public results?

We hope to produce a Draft Standard somewhere in '93, and a final draft for approval in '95. Of course, the whole proceeding is public and anybody can be an observer and anybody can be a member too; but you can only be voting under ISO rules if you are a national representative and under ANSI rules if you are a company rep. If you are a one man company then that's fine; the rule is there to stop IBM, AT&T and so on loading up the committee.

Several members are making it their business to inform people about what is going on so, although the first public draft is in '93, this isn't a secret process by any stretch of the imagination. I think there are 280 members that get all the written material, and 70-odd people present at a meeting who report home.

Some languages, such as Borland's Turbo Pascal, provide a* `typeof` *mechanism, by which you can obtain type information from classes at run-time. Are there any plans to incorporate such a mechanism into C++?

We are looking at it. There's a proposal coming out from Hewlett-Packard, and I've also worked on it - that's what's reflected in my book *(please see* The C++ Programming Language, *2nd Edition, pub Addison Wesley 1991 - Ed).* It's top of the list of things that we are considering. I guess that something will be approved along the lines described in my book.

There are three issues that are important: what is the language level interface of the type information?, what type information is guaranteed to be available? and how do you attach more information if you need it?

One thing that can happen, if you don't get this right, is that you load every application up with a couple of megabytes of type information. That must not happen. We have the principle in C++ of what you don't use you don't pay for. I would not like to see an application where if you didn't use type information you got an extra byte anywhere - and I think we can do that.

The actual interface is heavily biased towards the idea of a safe cast, as opposed to a `typeof` operator, because it's easier to get the correspondence between the static type information and the dynamic type information right with the safe cast. It also means that you don't first have to ask what the type is and then cast - as in the traditional approach. So I think it is much better to have a pointer cast operator than a `typeof` operator or a `kindof` operator - you get shorter, safer code with fewer bugs.

Are you rejecting anything that adds an overhead to C++?

That was not quite what I said. I said you should only pay for it if you used it. If you

use a little bit, it should cost you a little bit, and if you used a lot, you pay out in loads. And I guarantee if you want everything in type information it will cost you a lot. It does in every language and there are fundamental reasons for it.

Libraries

Do you have any plans to expand C++'s Standard Library?

Yes. The Standards committee is working - and has worked for some time - on ways of specifying a library, using the I/O stream library and the string library as guinea pigs. They are now expanding their research to include simple container classes like the vector, associative array (aka 'map' or 'dictionary'), maybe bitmap - trying to identify a nucleus of things that can be made available. I hope this is feasible without tying down everybody to the same architecture.

One fundamental aspect of our approach is that we're trying to build concrete types that are as optimum in time and space as you can get, and then looking at more abstract types built on them to provide generalisations. Contrast this with the Smalltalk way of building everything on top of a single base class, which imposes an overhead on everybody. It's another case of what I just said: what you don't use you don't pay for. If you want to use an associative array and I want to use a string, I don't want the overhead that is necessary for the associative array and you don't want the overhead that is necessary for a string. We should be able to separate the world in this way.

So you are completely against 'cosmic' libraries?

Oh yes. I don't think a universal data type has any place in C++. It's one of those things... If you have a universal base class in your design you are stuck. The idea of a universal base class is poison. It means that everything has to go towards that class, and everybody has to agree to get anything done. Yes, you get that standard library, and that will give people a leg-up, but you can't get further.

Smalltalk-style libraries have their place, but they're very limited. If you have Smalltalk-style libraries, and I don't, I can co-operate with anybody else that doesn't. You can't co-operate with anybody that either agrees with you on your philosophy - because they have their own universal class - or with somebody that agrees with me - because they don't want your universal class. Therefore you have to set out to take over the world. The world is too big for that.

This is quite a strong position that you're taking...

Yes. I have been saying it for years. People don't want to believe me, because they want the One True Religion. It may be Smalltalk, or some derivative of LISP, and these days Eiffel seems to be very popular among preachers. If people learn from preachers then they get into bad habits. They write programs that are big, clumsy and that run slowly, and they get very expensive failures when they can't deliver. They also get concepts they can't master. Maybe they could master them next year - that's fine - but take your time, and grow up along the way.

> ***Someone must have sold him the idea of OOP as a solution to everything - one of these flaming religious types***

If you want Smalltalk, for heavens sake use Smalltalk - it's the best Smalltalk around. If I had wanted to build a Smalltalk implementation then I could have built a much better one than C++. It is not a Smalltalk implementation. Sorry if I emphasise it so much, but I keep on hearing this and I don't want to.

OOP

I'm beginning to hear programmers make remarks along the lines of: 'OOP and C++ are all very well, but marketing people and journalists have been hyping them something rotten... They don't offer the ordinary working programmer anything very much.' How do you react to such a remark?

If someone said that, I'd think that someone had sold him the idea of object oriented programming as a solution to everything... in particular that you have to make everything derive from a single base class, or maybe make all your functions virtual - one of these flaming religious types. If he has run into that, and he thinks that is what C++ is supposed to be, then it's an example of a thing that I worry about quite a lot: bad teaching.

I recommend that you start slowly. Start using C++ as a better C, take advantage of the type-checking and the better notation, use a little bit of data abstraction, maybe use some libraries that do something useful for you, say X window interfaces, or mathematics. Pick up something that fits; go slowly from ordinary C to more type-safe C with a little bit of data abstraction. And then later, when you have some need for it, start using a bit of class hierarchies where they fit. This idea that you have to go all object-oriented to get benefits is hype, and it doesn't come from me and my friends. It comes from people who are trying to over-sell various other languages.

Eiffel influences

You mentioned the 'E-word' - Eiffel - a moment ago. It seems to me that a lot of things that were originally in Eiffel have appeared in C++ over the years -

- and *vice versa* -

- and vice versa. One thing that Bertrand Meyer/Eiffel is very hot on, which is conspicuously absent from C++ , is language support for preconditions/postconditions etc. Do you 1) think that these things are useful, and 2) if yes, would you consider incorporating them into C++?

Well, 1) they are useful and 2) they don't belong in the language. If you just want pre- and post- conditions, you just use either an assert macro or template. You don't need the language support to get most of the benefits. Secondly, if you need support in terms of verification technology and such, you can use an annotative language that uses a much stronger technology than you can build into a compiler. Then you get into something like A++, which has a verification system which is stronger than you get from something like Eiffel. But it is not in the language, it is a separate tool, like a design tool.

The Eiffel crowd in general, and Bertrand Meyer in particular, has a tendency to say that certain features are not only 'good' but 'essential' for writing software, and that certain features are not only 'not very important' but they are 'bad'. This is very much against the way I think. You can write reliable software in anything - even assembler. It's just a matter of convenience and

ease of modification once you have done it. What we are dealing with, in my opinion, is a trade-off between what the language provides, what you have to provide by tools and what you have to do for yourself. I have voted with my feet several times for putting things in the language that, for instance, the C crowd and the Pascal crowd said wasn't necessary. But it's useful - it's proven useful in real use.

We always promised that C++ would grow. If you look at my paper *What is Object-oriented Programming* from '86 it says that there are problems with C++. 'We don't have templates (as they are now called). We'll have to put in something to fake them - it will probably be better to have language support, I think that we'll get them some day.' It's the same for multiple inheritance and exceptions, and we are now looking at run-time type identification. If you read the first paper from back in '81 on 'C with classes', it said that I was looking at a range of possible extensions, and listed some things we might get when we understood them a bit better. It's interesting that multiple inheritance was on the list back in '81. It took me until '87 to figure out how to do it in a way that fitted.

Yes, C++ does not have everything. I don't think it should have everything. But it's not a static entity. And whatever the Eiffel people say, Eiffel is one of the fastest changing languages around. They realise it. Just for some reason - probably marketing - they don't admit it out loud the way we do.

In fact, if you look in my book there's a rather neat template for doing assertions. It doesn't give you everything you can get in Eiffel, but Eiffel doesn't give you everything you get in an annotation system like A++.

How do you respond to criticisms of C++ being 'impure'?

It is not right to be pure. It is right to serve your own and others' needs. Diversity of approaches has been shown to work. There's not just one right way, and anyway I have a problem with the word 'pure', because it makes me think of Stormtroopers. I have a problem with the people who think there is one right way for evermore. They can't get along unless everybody does it that one right way. It's a fundamental flaw in a language.

About your new book...

The new version of The C++ Programming Language struck me as quite different from the previous edition: more readable and less academic. Is this deliberate?

Writing is the only thing you learn by [w]rote... so some changes you can attribute simply to reading the thing over again. But the conception of the two books is the same. They both try to give people enough information to complete significant projects successfully, based on the experience I have had of problems that people really run into. In '85 one set of problems was bothering people, and now, six years later, there are different problems. Some things are not so much a problem any more, because people have learnt to cope. Other things are more problematical, because users have come in with more various backgrounds, and some side issues have become much more significant. More people are writing C++ programs. In '85 we were dealing primarily with teaching programmers working in small groups. Today, there are many many groups, some of them large, and this dictates the shift in the discussion.

But if the book's more readable then that's very nice, because I worked like mad.

Borland's argument is that nobody would ever want that feature on a system that wasn't mad

MS C++

What do you know about Microsoft's implementation of exception handling in C V7.0?

Nothing. Well, that's an overstatement, but a close approximation. I know they were dabbling with resumption at some point, but I don't know if they still are. So 'nothing' is a fair answer. But don't take it as being criticism or comment or anything, it's just: yes, I've talked to some of the guys who worked with it but I don't know what they actually went and did. I certainly don't have any opinions on it.

Do you think the release of MS C++ will increase acceptance of it as a 'better C'?

There's two questions there, especially as 'a better C' is a buzz-word in the C++ world. Yes, I think that Microsoft coming out will help the acceptance of C++, though with Zortech and Borland already out it will be another stage in acceptance rather than some revolution.

Microsoft will, of course, bring in new programmers. My recommendation is that new programmers should start with a 'better C' level of C++ until they have had time to learn more. So if, as one would expect,

Problems of C's declarative syntax

The implicit `int` problem is show by:

```
int f(const T);
```

Is `T` a type name (so that `f` takes a argument of type `const T`) or is `T` a variable name (so that `f` has an argument call `T` of type `const int`)?

Had `int` been required for all `ints`, the user would have had to write:

```
int f (const int T);
```

in the second case, and there would have been no opportunity for confusion.

The following is hard to read:

```
int (*v[10])(char);
```

which is an array of 10 pointers to functions taking a `char` argument and returning an `int`.

The problem is that to interpret this is that you must read the declaration line 'inside out'. What would have been much easier to read and write would have been a linear notation (like the English above). For example

```
v[10]->(char) int        // full
```

or

```
int v[10]->(char)        // halfway
```

The latter is a compatible C extension discovered circa 1981 documented by Rani Sethi in SP&E paper.

Figure 1 - C's declarative syntax

Microsoft C++ brings in a large number of new C++ programmers, then they will be a lot of 'better C' style C++ programmers. That is exactly how I want it. Then you wait six months to a year, you'll find quite a lot of 'better C and data abstraction' style programmers. Wait the same time again, you'll find there are a lot more programmers doing much better abstraction code, much better class hierarchies, and much 'typesafer' code. I think this is an important difference between C++ and some other approaches.

You don't expect things to happen in one step. I strongly hope that the new C++ programmers will read books like mine and others that are cautious. On the other hand, a lot of books just try to say: 'Well, we need to have purity of approach. Go overboard and do everything at once. If you can't figure it out then you shouldn't be licensed to write a program.' There's lot's of good books and even more bad books, and I just hope that people will be a bit careful. I also hope that journalists won't declare C++ dead because people will - as they should - write better C in the beginning.

But won't people just write better C forever, and never write real C++?

Oh, they never stop learning and improving. My experience is people do not stop with 'better C'. They slide along very nicely, provided they have half a reasonable educational experience and materials. It takes a little time, but it always happens.

But going in at the deep end - saying you can't write any C++ until you can write some major class hierarchy - is just asking for failures. We've seen such failures in other languages, but not to my knowledge on a big scale with C++. Caution is what keeps it that way.

What if you are a virgin C++ programmer coming onto a mature project where many classes already exist...

It is so much easier to use a class than to design it.

So you're not discouraging novices from using class libraries?

Oh no. Use the class libraries. Use the libraries that you can either buy or that are used in your project. The simple example is: using a string class is far easier than using the C primitives for string manipulation. Writing a string class is almost a rite of passage. Everybody has to write two to try and come to grips with the abstraction mechanisms in C++. So write your string classes - then go and use somebody else's.

A Maritime Parable

With the introduction of templates and exception handling, do you see the C++ language as essentially stable or are there other major issues still to be addressed?

> ***C has in some sense stopped growing. The pipeline is not being primed with new C stuff, it's being primed with C++***

There's two aspects to stability. One is: to stop growing. The other thing is if old code breaks. One of our official aims in the ANSI and ISO process is to make sure that as little as possible code breaks - and preferably none. If something is indisputably a legal C++ program, it shouldn't break. It's as simple as that.

You can grow and still maintain stability by adding things compatibly. I think the major features are in place, but people propose new features very, very often and with great fervour. We're looking at a lot of things. Top of the list is the run-time type identification issue.

I often tell people about the good ship Varsa. This was built in Sweden during the 17th century to be the biggest and the best and the most beautiful battleship in the Swedish navy. But somewhere during construction came the notice that somebody else was building bigger battleships. And it was observed that if this wonderful Swedish battleship came up against a battleship with two gun decks, the Swedish ship would be on the bottom with a lot of holes in it very fast. So they decided to make a quick correction to this problem and added another gun deck. The ship now got even more beautiful, as there was room for even more sculptures. The designer was literally driven to his grave out of worry about what was happening. The ship made it half-way across Stockholm harbour and keeled over and sank killing 50 people. Which is why you can go to Stockholm today and see it, and it's a wonderful sight to behold - much prettier than if it had gone to the bottom the traditional way.

There are limits to what you can do to a language like C++, and we're very careful about that. I think that if all people want all their favourite features from all the languages they have ever used, they should spare a thought for the good ship Varsa.

Borland's bits

What is your view of Borland's DDVT extension to the language to cope with message handling in the Windows GUI environment?

There's two things that have to be remembered. You have to remember both of them, because either one by itself doesn't sound very good. First: I don't like extensions. Second: sometimes they are necessary. When you have architecture as weird or as warped as the original PC architecture, you probably need some extensions. Some of them are relatively benign. `near` and `far` may be considered perverted, but it does not harm the fundamental structure of the language. On a Cray you might have very different kinds of extensions. Sometimes they're the only way of getting a general-purpose language to take advantage of special architectural or operating system features. But it's always a judgement: When is something a warp of the language to take advantage of a particularly warty environment or architecture, and when is it a general-purpose extension that everybody has to live with?

I know Borland's argument, and that is that the feature is there simply to fit into a weird environment, and nobody would ever want that feature on a system that wasn't that mad. Therefore it's no problem. I actually don't know enough about the case in hand to see if their argument is true. It could be a marketing ploy, and it could be perfectly true. It's not my job to judge that one.

Do you think that C++ needs some kind of mechanism - not necessarily DDVTs - to handle GUI programming more efficiently?

You can design your GUIs so that you don't need that kind of extension. I don't think that the language should have features simply to support bad designs. If somebody can prove that it's not a bad design, but a

fundamental necessity of GUIs then yes, of course the language should be extended to do it, but I doubt very much that is the case.

You've said on record that C's declarative syntax was an experiment that failed. Did you have an alternative in mind?

Oh yes. I can give you off-line an example of this (Figure 1) that looks really weird. But don't get anybody's hope up. I tried to get it done years ago - and failed.

How do you feel about the large growth projected for C++? Your own estimates of growth seem to be quite conservative, compared with the results of the .EXE reader survey.

If I didn't round down my estimates heavily they would be hype. The take-up may well be larger, but let's wait and see. My feeling is that C has in some sense stopped growing. People don't build C tools any more, people don't build C compilers any more. There's still some backlog coming out, but the pipeline is not being primed with new C stuff, it's being primed with C++.

I doubt that that many people will actually be doing C++ in half a year, but they will in a year. Probably in six months all the new compilers will be C++, meaning that if people want to use the latest compilers or the best code generators and such, they'll be using C++. A year from now they will have stopped using the C compatibility switches, and they will have verified for themselves that there are benefits in the stronger type checking. They will all be looking at the data abstraction and inheritance mechanisms, hopefully with sensible suspicion. And a lot of them will be starting to use it.

> ***I don't think a universal data type has any place in C++ - the idea is poison***

I just hope they can benefit from other people's experiences. Of course, I recommend my book, but there's other good books too. My rule is: look to see if there is a history section, and look to see if there is a lot of hype about object-oriented programming at the beginning; and if there's no history and there's object-oriented hype - don't touch it!

Of course there's people with other temperaments than mine, who have more courage (or foolhardiness) than I have. I prefer to see people moving relatively slowly and manageably. Don't rush to the other end, because by and large you'll get there in the same time anyway. In my mind, doing a sprint and then having to walk because you are puffed is not a good strategy. Some people like it that way - but they'd do it anyway, whatever I said.

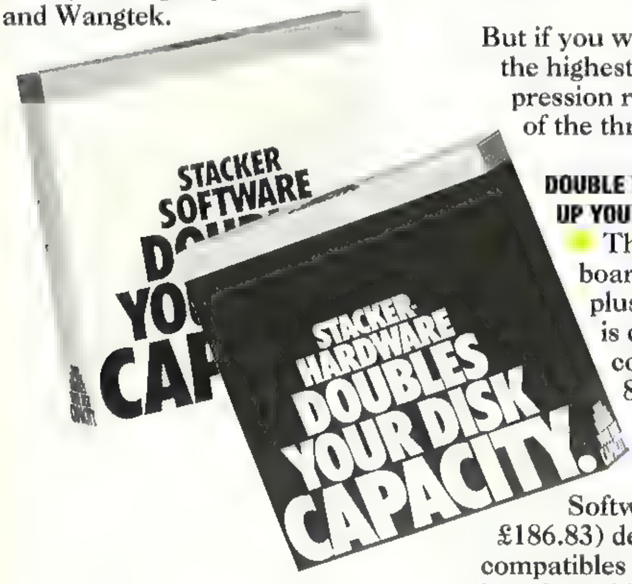
EXE

Many thanks to Dr Stroustrup for sparing the time to give this interview, and to the folks at the European C++ User Group for setting it up. Call ECUG on 071 2535121.

DOUBLE YOUR HARD DISK WITH STACKER.

Stacker instantly doubles the size of your existing hard disk - so 20Mb becomes 40Mb, 40Mb becomes 80Mb, 80Mb becomes 160Mb, and so on. And at no risk to your valuable data.

Stacker is 100% compatible with MS.DOS, DR. DOS & Windows, it installs in an instant and is totally invisible in operation. It's the market leader in the USA and the industry standard used by Peter Norton Computing, Central Point Software, Exabyte, Megatape, Tecmar, and Wangtek.



from **£99**

£99 for STACKER SOFTWARE

Double your hard disk instantly.
Four versions are now available starting with...

- Software only at £99 plus VAT (ie. £116.33) which is perfect for portables, laptops and notebooks, or where there are no available expansion slots.

But if you want small driver size and the highest possible speed and compression ratio then just choose one of the three hardware versions.

DOUBLE YOUR HARD DISK *AND* SPEED UP YOUR XT/AT/386/MCA PC

- The XT/8 with Coprocessor board and Software at £129 plus VAT (ie. £151.58) which is designed for IBM PCs and compatibles (ie. Intel 8088/8086 based machines).
- The AT/16 with Coprocessor board and Software at £159 plus VAT (ie. £186.83) developed for IBM PCs and compatibles (ie. Intel 286/386/486 based machines).
- And the MC/16 with Coprocessor board and Software at £189 plus VAT (ie. £222.08) which is specially produced for the Microchannel environment.

"Easy to install, massive performance benefits without hardware outlay, completely transparent."

"As a simple way of upgrading a system without hassle, it is hard to imagine a more useful program."
— Personal Computer Magazine

YOUR NO NONSENSE GUARANTEE FROM TKH

Risk free purchasing with free technical support.
TKH International specialises in powerful products at very competitive prices. We deliver free of charge in England, Scotland and Wales. And provide FREE instant Technical Support through our Hotline Experts. All our products come with a No-nonsense, 30-day Money Back Guarantee. So, if you're not completely satisfied, simply send the product back and we'll refund your money.

HOW TO ORDER

Simply call

081-995-4500

to use your Access, Visa or American Express Card.
Or, send a Cheque or Company Purchase Order made out to TKH International at
TKH INTERNATIONAL,
FREEPOST, LONDON W4 5BR

TKH INTERNATIONAL

Please quote ref GS1

CIRCLE NO. 560

Life without Huffman

The Huffman approach to data compression is the best known, but it can be hard to implement. Crosbie Fitch has stumbled upon an alternative.

To date, Huffman Code stands as the most efficient, direct method of encoding an arbitrary number of tokens of varying frequency into variable length binary sequences. However, it is not particularly easy to implement. For those of you that would prefer something easier to digest as well as implement, while being of comparable efficiency, I'd like to introduce you to Unary Prefix Code (UP Code).

Just like sorting algorithms, where there is still a place for the bubble sort, I think it would be a pity if we forgot the less efficient coding methods. They can be quite educational, not to say useful in some situations.

Like many programmers I've done my share of re-inventing the wheel. Indeed, most programmers usually recognise when a discovery is liable to have already been discovered and is thus a mere re-invention. Of course, there are times when things go the other way; a real discovery is dismissed by its author until it's recognised to be original (nearly always by someone else). This was the case with Huffman code (see *Scientific American* September 1991, p27). David A Huffman developed the method in response to a term paper problem set by his professor, who surprised him upon revealing that its solution had, until then, not been perfected.

I discovered Unary Prefix Code in ignorance of Huffman Code or the idea of using binary digits as directions for traversing a coding tree. For those of you that don't know, a detailed description of how to implement Huffman Code is in the book *Algorithms in C* by Robert Sedgewick (pub Addison-Wesley, ISBN 0-201-51425-7). When I heard about Huffman Code I thought that it was what I'd re-invented. No such luck. I'd only discovered a slightly slacker coding scheme that I suspect was old hat to David Huffman and his fellow academics. Nevertheless, it does have its merits and, though not optimal, has the appeal of being simple to understand and implement.

How it works

The idea of all such encoding is that given *N* different items occurring *Fn* times in a sequence, how can you represent each item as a variable length binary code so as to use the fewest number of binary digits in the encoded sequence? Obviously the items that occur most are given the shorter codes, but the question is: What codes should be used and precisely how should they be allocated? A typical application is that of compressing an 8-bit text file where only a few of 256 possible characters are used, and furthermore, have markedly different frequencies ('E' occurs far more than 'Z').

Unary Prefix Code is as it's called; giving a binary number a unary most-significant-digit. Moreover, the binary number, or suffix, is of variable length.

Now, being unary, the prefix enhances the range of lengths of the code and so becomes ideal in situations where there may be a wide variation in frequency. It allows total flexibility of code length, where with a fixed length binary prefix you sacrifice being able to have very short codes for the lesser benefit of having shorter, long codes.

One example of a useful binary prefix code is where the msb of a byte determines whether the following value is stored in one byte or two. This allows values of 0-127 in 8 bits and values of 128-32895 in 16 bits.

So, in summary, UP code uses the prefix to encode information about which group of items are represented by the suffix and how many items are in the group.

Unary

You're probably wondering how unary numbers can be implemented on a binary processor. Well, you can, in the same way that you can implement a series of decimal digits in hexadecimal, where you can consider every digit decimal until you find one greater than 9. Unary is quite similar. Every '0' bit is unary until you find a '1' (reading from left to right). Thus 0,1,2,3 in unary is '','0','00','000' and '1','01','001','0001' in BCU (Binary Coded Unary) - see Figure 1. I won't go into philosophical discussions here about whether zero can be represented in unary!

Let me demonstrate with a very simple example, the 11 letter word: 'ABRACADABRA'. 'A' occurs five times, 'B' & 'R' twice and 'C' & 'D' once. See Figure 4.

In 8-bit ASCII this would occupy 88 bits (we'll presume the length does not need to be encoded). We could encode the letters as a three bit character set and reduce the size to 33 bits. In BCU we could code 'A'=1, 'B'=01, 'R'=001, 'C'=0001, and 'D'=0000. This would reduce the size to (1x5+2x2+3x2+4x1+5x1) = (5+4+6+4+5) = 24 bits. Notice that we don't need to termi-

Decimal	Binary (3-bit)	Unary	Binary Coded Unary	Unary Prefixed Binary
0	000		1	10
1	100	0	01	11
2	010	00	001	0100
3	110	000	0001	0110
4	001	0000	00001	0101
5	101	00000	000001	0111
6	011	000000	0000001	001000

NB: Reading left to right

Figure 1 - Binary and unary compared

```
static int getupcode(int occ[],int n,
                               int ul[])
/* ENTRY >> occ[]:
   Number of occurrences of n items,
   sorted in descending order
   EXIT  >>  ul[]:
   Number of bits in suffix of each
   of (returned) prefix codes */
{
  int total, /* Total occurrence of
                current and as yet
                unallocated items */
      item,  /* Item */
      oc,    /* Occurrence of item(s)
                to be allocated to
                current unary prefix */
      u,     /* Index of current unary
                   prefix */
      i;     /* General index */
 /* Get total occurrence of all items */
 for (total=0,i=n; i--; total+=occ[i])
  ul[i]=0; /* Reset UP lengths */

 /* For each prefix, until all processed
   (maintain total of unallocated ocs) */
 for (u=item=0; item<n; ++u,total-=oc)
   /* Extend suffix until ocs exceed half
      remainder (oc>(total-oc)/2) */
   for (oc=occ[item++]; oc*3<=total;
                               ++ul[u])
     /* Accumulate ocs of other half of
        items in extended, suffix group */
     for (i=1<<ul[u]; i--;
                    oc+=occ[item++]);

 /* Combine final suffixes
                  of equal length */
 while (u>1 && ul[u-1]==ul[u-2])
 /* decrementing number of prefixes and
   doubling items in penultimate
   suffix group */
   ++ul[--u-1];

 /* Return number of unary prefixes */
 return(u); }
```

Figure 2 - The C function getupcode()

```
/* Demonstration of Unary Prefix Code */
/* by Crosbie Fitch */

#include <stdio.h>

#define DIM(A) (sizeof(A)/sizeof(*(A)))

static int item_occ[]={11,6,5,5,4,4,3,3,
                3,3,2,2,2,2,2,1,1,1};

#define MAXCODEN DIM(item_occ)

static int getupcode(int [],int,int []);

int main(void)
{
  static int upcode[MAXCODEN],
             upcoden;
  int i,j;

  upcoden = getupcode(item_occ,MAXCODEN,
                                 upcode);

  for (i=0; i<upcoden; ++i)
  {
    for (j=i; j--; )
      putchar('0');
    if (i<upcoden-1)
      putchar('1');
    for (j=upcode[i]; j--; )
      putchar('x');
    putchar('\n');
  }

  return(0);
}
```

Figure 3 - A C program to test getupcode()

nate the code for 'D' since the decoder will know that unary 4 is the largest code used. In UP code we'd use 'A'=1, 'B','R','C'&'D'=0xx. This gives a size of (1x5+3x(2+2+1+1)) = (5+3x6) = 23 bits. Again, the last code's prefix doesn't need to be terminated. In Huffman code we'd use 'A'=1, 'B'=00, 'R'=010, 'C'=0110, 'D'=0111. This gives a size of (1x5+2x2+3x2+4x1+4x1) = (5+4+6+4+4) = 23 bits.

UP code often gives very close performance to that of Huffman code, in fact it has to be a carefully contrived example that displays a marked difference.

How to do it

Although I haven't got the space for details about how to write routines to compress and decompress files using UP Code, I will provide you with the critical routine to calculate the number and lengths of UP codes. It is fairly straightforward to pursue the idea further from this point.

Figure 2 is of the C function `getupcode()`. This takes as input a sorted array of occurrence counts. It returns an array of suffix lengths. Figure 3 is of a demonstration program which will output a list of UP codes resulting from a particular list of occurrence counts. Given the ABRACADABRA example, {5,2,2,1,1}, it would produce:

```
1
0xx
```

The zeros and ones make up the unary prefix and the xs indicate the binary suffix. Figure 3 is based on the frequency counts in the example from the Sedgewick book. It produces the output shown in the first column of Figure 5.

Note that the codes are allocated in unary order to the items in sorted occurrence count order. Thus the first item, occurring 11 times, is given the code 100; the 2nd of 6 times, 101; 3rd of 5, 110; and so on. The encoded text would start off 01011000110110...

The reason UP code isn't as optimal as Huffman code is that UP code restricts the size of suffix groups to be a power of two. Huffman code can have totally irregular suffixing. Thus, instead of having to have four 3-bit codes, as in '1xx', Huffman code can use codes 100, 101xxx and 11x, three 3-bit codes, using the fourth as a prefix for a longer group.

I'd recommend UP code for high speed situations where there is no requirement for optimal compression of large files. To decode, all it requires are simple bit shift operations that increment an index into an array of suffix lengths and corresponding indices to the original items. Incidentally, if speed is paramount, it might be preferable to leave the termination bit on the last prefix.

A trick that might help decoding is based on the idea of first extracting the unary prefix, using the formula

```
Prefix =
      UPCode & (~UPCode + 1)
```

For example, if the UPCode is 001xx, which is stored 'back to front' as bbbxx100, then we have:

```
Prefix
      = bbbxx100 & (BBBXX011+1)
      = bbbxx100 & BBBXX100
      = 00000100
```

The prefix may then easily be determined by comparison or shifting to zero, eg

```
while (x>>=1) ++count;
```

Alternatively, if you have memory to spare, you can create a look-up table of values for each possible prefix signature.

Item	ASCII	3 bits	BCU	UP Code	Huffman Code
A = 5 x	8	3	1	1	1
B = 2 x	8	3	2	3	2
R = 2 x	8	3	3	3	3
C = 1 x	8	3	4	3	4
D = 1 x	8	3	5	3	4
Total bits	88	33	24	23	23

Figure 4 - Comparison of encoding ABRACADABRA

Test String:
A SIMPLE STRING TO BE ENCODED USING A MINIMAL NUMBER OF BITS

UP Code	Occurrences	Letters
1xx	11,6,5,5	_,I,E,N
01xx	4,4,3,3	M,S,A,B
001xx	3,3,2,2	O,T,D,G
0001x	2,2	L,R
00001	2	U
000001x	1,1	C,F
000000	1	P

Figure 5 - UP Code details of a test string

Prologue data

One area in which UP Code is more efficient than Huffman Code is that of encoding information about the encoding key. Both methods also have to supply details of what each code represents, but since that is the same for both, I won't describe it here.

Huffman Code must supply details of the binary coding tree. Since it is a binary tree it can be represented in binary too, using a bit for each node. A zero bit indicates a leaf and a one bit indicates a branch. The coding tree needed to encode "A SIMPLE STRING..." requires 18 bits for the leaves and 17 bits for the branches; a total of 35 bits. Note that this tree will also reveal how many different codes there are.

UP Code needs only to supply the suffix lengths to each unary prefix. An initial value declares the number of prefixes (-1) which follow in BCU. I suggest the use of regular Unary Prefixed Binary (UPB) for this first value. UPB is an equal number of bits used for both unary prefix and binary suffix, and so progresses: 0..1=1x, 2..5=01xx, 6..13=001xxx, etc. Figure 6 shows how the UP Code list has been encoded. This gives a length of 21 bits.

If *N* is the number of different items encoded, then the encoding information size *E* is constant for Huffman Code at $E = 2N-1$ bits. Unfortunately, there is no simple formula for that of UP code (using my method). Even so, I estimate that *E* varies between $3+Log_2(N)$ bits and $3+N+2Log2(N)$ bits. With $N = 18$, as in the preceding example, *E* could actually vary from between 9 to 28 bits. Anyway, if we're dealing with small file sizes, the bits saved in the header of UP Code can often make up for the bits lost in the non-optimal coding. If we had the same frequency distribution as in the example, it would take a file size of about 840 items (as opposed to 60) before Huffman Code becomes more compact (a difference of one bit every 60 items).

One final point to mention: you may decide you prefer BCU with 1s as digits and 0s as terminators. Thus instead of 1, 01, 001 and 0001 you have 0, 10, 110 and 1110. This way the sum of the digits accords with the value. My justification for using it the other way around is to continue with the pattern of the highest valued digit being the base-1. Decimal 0Ah - 1 = 9, octal 8 - 1 = 7, binary is 2 - 1 = 1, so I thought unary should be 1 - 1 = 0.

UP Code prefixes	Encoded
6 (n-1)	001000
1xx	001
01xx	001
001xx	001
0001x	01
00001	1
000001x	01
000000	1

Figure 6 - Encoding the UP Code list

I hope this has been of interest. If I haven't persuaded anyone of the merits of Unary Prefix Code, I'd be happy to think that at least some people have had the idea knocked out of them that Huffman Code is the only useful, variable length, binary encoding scheme.

EXE

Crosbie Fitch has worked as a C programmer for the Institute of Manpower Studies for the last six years. He is a fan of RISC computers, such as the Acorn Archimedes, and contributes to various Acorn related journals.

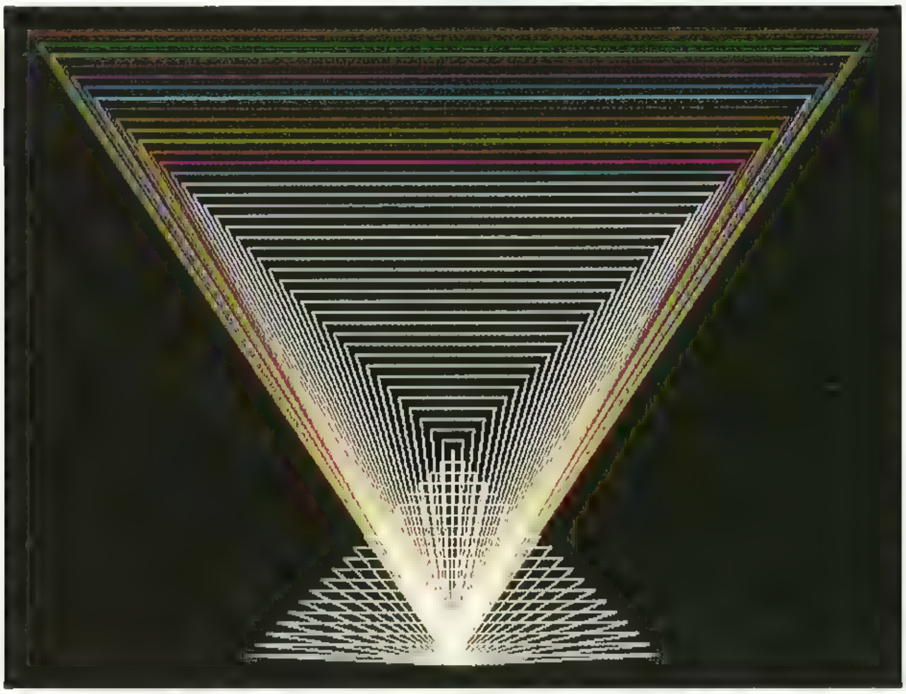
The Fastest Draw

Which C compiler is the best for writing DOS graphics applications? Cliff Saran assesses five contenders.

If you want to produce respectable DOS graphics on a shoe-string budget, check what your compiler has to offer before considering third party libraries. This month I have been investigating the strengths and weaknesses of the five graphics libraries bundled with Borland C++ V3.0, Microsoft C V6.0, Topspeed C V3.0, Watcom C V8.5 and Zortech C++ V3.0.

How different are these libraries? How fast are they and what's it like to write code using them? The short answer is that they all offer similar capabilities. What's more, three of these libraries (from Microsoft, Topspeed and Watcom) provide more or less source code compatibility with each other. To make life easier, I'll refer to this trio as the Microsoft compatible libraries. In Figure 1, I have compiled a list of features supported by the five libraries.

When attempting to produce a meaningful comparison between several libraries it is impossible to provide a definitive set of tests that reflects their complete functionality. I have opted for testing the subset which I consider to be the most likely graphics functions that an 'average punter' would want to use. There is obviously a strong element of personal judgement in this choice of functions, but I believe that my benchmark is a fair indication of the performance and functionality provided by these five libraries. The results of the benchmarks are given in Figure 2. For a description of the benchmarks see the separate box.



ADVANCED BUSINESS GRAPHICS

Borland BGI

The function call required to initialise Borland's BGI engine is a little intimidating.

```
void initgraph
(
  int far *graphdriver,
  int far *graphmode,
  char far *pathtodriver
)
```

This complexity is due to the immense versatility that Borland has built into the library ie it is possible to load your own custom graphics card driver into the BGI library (the BGI driver spec is available from Borland). Although Borland provides drivers for the standard CGA, EGA and VGA graphics modes, there are several SVGA drivers available in the public domain from such places as the Borland conferences on CIX and CompuServe, and also various commercial third party offerings.

Underlying BGI is the concept of a hypothetical graphics cursor which moves in response to various calls to the library. Lines can be specified either uniquely (ie providing both end-points) or from the current cursor position. Borland allows you to change the appearance of a line in two ways. You can specify both the thickness of a line and the way in which it is drawn (ie Solid or Dashed). There are four built-in line patterns and you can also specify one of your own. When drawing shapes, such as rectangles and polygons, Borland uses the current settings for both the colour and the line style so the settings remain in force until they are changed.

Circular shapes (ie ellipses) are defined by specifying a centre and a radius. Arcs and sectors additionally require a Start and End angle in degrees. Borland's approach to defining these shapes is extremely 'user-friendly' and it maps easily onto the way we usually picture them mathematically (see Figure 3).

There are several built-in fonts which Borland provide for writing to the graphics screen. The Stroke-based fonts are character-sets specified with vectors, making them extremely easy to scale. The bitmap-based fonts are aesthetically better to look at than stroked-based characters, but these characters cannot easily be scaled. Borland provides one bitmap font and 10 stroked fonts. User-defined stroke-based characters can

YOUR ONE-STOP SOURCE FOR THE LATEST TOOLS AT THE BEST PRICES

PRE-SALE CONSULTING

POST SALE SUPPORT

NEW & NOTEWORTHY

Price
- ABC Flowcharter v1.1 for Windows £139
  Priced to go. Get yours now!
- Baler v5.1 ..... £99
  Only a few left, priced to go!
- Borland C++ v3.0 & App. Frameworks .. £289
  Unsurpassed DOS & Windows devlpt envir with ready-made user interface objects.
- BRIEFFor C++ £75
- Cardinal 9600 MNP-5 V.32 Modem £265
  9600 baud, MNP-5, V.32 operation, V.42 compatible INCREDIBLE VALUE!
- Corel Draw v2.0 CD-ROM £319
- GRASP with ARTOOLS v4.0 ..... £159
  Complete control over all aspects of Graphics Animation and Presentation.
- Laplink Pro. v4.0 ..... £65
  Limited time offer. Get yours while supplies last!
- MS OS/2 Presentation Mgr Toolkit v1.2 £239
  Priced to move, only a few remaining.
- MS QuickC/Windows & Windows SDK £309
- Multi-Edit v6.0 ..... £65
  New Version!
- Object Vision v2.0 ..... £69
  Create full-fledged Windows applications visually, without programming.
- Paradox v3.5 with DeskTop £145
  Limited time offer 'til 31/03/92
- Photofinish £45
- Pizazz Plus v3.0 ..... £49
  Screen Capture, Print & Export Utility A 'must' for Desktop Publishing!
- RPG II Development System v4.0 ..... £999
  New version.
- Show PArtner F/X v3.6 ..... £179
  Presentation and Animation. free Runtime
- SoundBlaster Pro £165
  Multimedia is not coming...it's here! Music, Speech, MIDI, CD-ROM Interface.
- Spontaneous Assembly v2.0 ..... £85
  Priced to go, only a few left!
- Stacker v2.0 ..... £69
  Double your hard disk capacity'
- Turbo C++ for Windows £75
  Includes Resource Workshop, Graphical tools speed up & simplify Windows devpt.
- TURBO VISION DEV TOOLKIT v2.0 .. £85

ARTIFICIAL INTELLIGENCE
- Arity Windows Toolkit v6.0 ..... £189
- Arity/Prolog Advanced Toolkit v6.0 ..... £85
- Arity/Prolog Compiler & Interpreter
- Arity/Prolog Interpreter v6.0 ..... £169
- Arity/Standard Prolog v6.0 ..... £55
- PC Scheme v3.03 ..... £59
- PC Scheme v3.03 ..... £59
- PDC Prolog v3.7 ..... £179
- VP-Expert v3.0 ..... £154

ASSEMBLERS & LINKERS
- ASM Tools v2.5 ..... £109
- Phar Lap 386\DOS Extender SDK v4.0 ..... £325
- PharLap 386\ASM/LinkLoc v4.0 ..... £799
- Sourcer 486 ..... £79
- Turbo Debugger & Tools v2.0 ..... £75

BASIC
- db/LIB Database Library £65
- GraphPak Professional v3.0 ..... £89
- Microsoft Visual Basic £89
- PowerBASIC £59
- QuickPak Pro v4.0 ..... £119
- Stay-Res Plus 4.0 ..... £125
- True Basic v3.02 ..... £59

BUSINESS
- ABC Flowcharter v1.1 for Windows £139
- Baler v5.1 Spreadsheet Compiler £99
- Grammatik v2.0 for Windows £45
- Micrografx Charisma v2.1 ..... £249
- ORG PLUS Advanced v6.0 ..... £49
- Super PRoject v2.0 for Windows £410
- Who-What-When v2.0 ..... £140

C AND C++
- Aztec C 86-p Professional System £125
- Borland C++ v3.0 ..... £195
- Borland C++ v3.0 & App. Frameworks £289
- C ASYNCH MANAGER v3.1 ..... £99
- C Communications Toolkit £85
- C TOOLS PLUS v6.0 ..... £79
- C Utility Library £129
- C++/Views £239
- C-ISAM £119
- C-terp v5 ..... £159
- C-Worthy v2.0 & cwArchitect £149
- CLEAR for C £89
- Codebase 4.5 ..... £125
- Essential Communications £169
- Greenleaf Comm++ £99
- Greenleaf Functions Library v4.02 ..... £119
- Greenleaf ViewComm v3.0 ..... £205
- Informix ESQL/C £295
- Intel 386/486 C Code Builder Kit £319
- Microsoft Quick C for Windows £85
- MPW C++ v3.2 ..... £149
- MS PRogrammer's Library (CD-ROM) £185
- NDP C-486 v3.2 ..... £745
- Paradox C Engine v2.0 ..... £75
- PC-Lint v5.0 ..... £159
- POWER TOOLS PLUS £79
- TopSpeed C Master edition £275
- TopSpeed C Professional v3.02 ..... £165
- TopSpeed C Professional edition £165
- TopSpeed C++ Standard £110
- TopSpeed C++ v3.02 ..... £50
- Turbo C TOOLS v6.0 ..... £79
- Turbo C++ 2nd Edition £49
- Turbo C++ for Windows £75
- TURBO VISION DEV. TOOLKIT v2.0 ..... £79
- Vitamin C v4.0 ..... £219
- WATCOM C 8.5 ..... £269
- WATCOM C 8.5/386 ..... £465
- WIN++ v2.1 ..... £139
- WindowsMaker v3.07 ..... £469
- Zinc Interface Library for DOS £139
- Zinc Interface Library for DOS & Win £199
- Zortech C++ Database Class Library £189
- Zortech C++ v3.0 for DOS, Win & OS/2 ..... £199

CADD
- AutoSketch v3.0 ..... £49
- Design CAD 3D £149
- DRAFIX CAD v2.0 for Windows £319
- Generic CADD 3D v1.1 ..... £169
- Generic CADD v6.0 ..... £219
- MathCad v3.0 for Windows £225

CASE
- Case PM for C and C++ £1,245
- Case W Corporate v3.1 ..... £620
- Case: W Standard Edition v3.0 ..... £309
- Compeditor I v3.11 ..... £129
- EasyCASE Plus v3.0 ..... £299
- EasyCASE Professional Pack v3.0 ..... £399
- System Architect £1,099

COBOL
- LPI COBOL Compiler for DOS £799
- RM/COBOL Development System v5.1 ..... £599
- RM/COBOL-85 Developer Pak + ... £935
- RM/COBOL-85 v5.1 Deployment Pack £75

COMMUNICATIONS
- Carbon Copy Plus v6.0 ..... £85
- Central Point Commute £55
- Crosstalk Communicator v2.0 ..... £29
- CrossTalk v1.7 for Windows £89
- Crosstalk Xvi £89
- Greenleaf CrossLib v3.2 ..... £185
- HyperACCESS/5 v2.0 ..... £49
- pcAnywhere IV v4.5 Host & Remote £89
- Procomm Plus v2.01 ..... £49
- Reflection 1 + for DOS £219
- Remote2 v2.1 Host & Caller £79
- SmarTerm 420 for Windows £125
- View 232 v1.1 ..... £109

DATABASE & FILE MANAGEMENT
- Advanced Revelation v2.1 ..... £539
- ALPHA/Four v2.0 ..... £249
- ALPHA/Four v2.0 Competitive Upgrade £75
- B-Tree Filer v5.2 ..... £79
- Btrieve v5.1 for DOS, Win or OS/2 ..... £299
- c-tree Plus v6.0 ..... £339
- Clarion Personal Developer v2.0 ..... £35
- Clarion Professional Developer v2.1 ..... £349
- CLEAR+ for dBASE £89
- Data Boss £345
- DataEase v4.5 ..... £399
- DataFlex Multiuser Devlpt System v2.3 ..... £510
- dbFast/Windows v1.5 ..... £205
- FairCom Toolbox Special Edition v2.0 ..... £419
- FUNCky LIB ..... £125 (PLEASE CHECK!)
- Genifer £175
- Informix 4GL Compiled £489
- Informix SQL £389
- Novell Xtrieve Plus v4.1 ..... £490
- Oracle Tools & Database v6.0 ..... £859
- Paradox C Engine v2.0 ..... £75
- Paradox v3.5 Competitive Upgrade £139
- PC/Focus v6.0 with Perspective Jr. £375
- R&R Code Generator £99
- R:BASE v3.1 ..... £359
- zyINDEX for Windows £185

DEBUGGERS
- Multiscope Debugger for DOS v1.02 ..... £89
- Multiscope Debugger for Windows £189
- Periscope Debuggers ..... CALL
- Turbo Debugger & Tools v2.0 ..... £75

DESKTOP PUBLISHING
- Adobe Type Manager for Windows £46
- Adobe Type Manager Plus for Windows £93
- Aldus Photstyler for Windows £399
- Facelift v1.2 for Windows £45
- Freedom of the Press Windows £179
- Go Script Plus £145
- Inset v2.2 ..... £79
- Legacy v2.0 ..... £109
- Microsoft Publisher for Windows £95
- OmniPage 386 v3.1 for Windows £325
- WordScan £69

DISK/DOS/BACKUP
- Above Utilities for Windows £45
- Back-It v4.1 ..... £59
- Central Point Backup v7.0 ..... £55
- Disk Technician Gold v1.04 ..... £79
- Master Tracks Pro v3.95 for Windows £199
- Norton Utilities v6.01 ..... £39
- PC Tools Deluxe v7.1 ..... £75
- PC-Kwik Powerpak v2.1 ..... £59
- Q-DOS 3 v1.5 ..... £35
- SpeedStor v6.0 ..... £49
- SpinRite II £49
- Super PC-Kwik v4.1 ..... £35
- Vopt £35
- XTree Pro Gold £75

EDITORS
- BRIEF v3.1 ..... £148
- Edix for DOS or OS/2 ..... £109
- EDT v5.0 ..... £165
- Epsilon for DOS, OS/2 or UNIX £159
- KEDIT v4.0 for DOS & OS/2 ..... £95
- Multi Edit Professional v6.0 ..... £99
- Qedit Advanced v2.1 ..... £39
- VERDIT Plus v3.5 ..... £99

FORTRAN
- Cosort £79
- FortLIB v6.0 ..... £149
- Fortran Toolkit 1 v5.2 ..... £65
- Fortran Toolkit 2 v5.2 ..... £99
- GRAFMATIC & PLOTMATIC £129
- Lahey F77L £355
- Lahey F77L v5.01 - EM/32 v4.02 ..... £595
- Lahey/Ergo OS/386 Operating System £235
- LPI FORTRAN v3.0 ..... £465
- RM/FORTRAN Development System £279
- Spindrift Utility Library (F77L) £69
- WATCOM Fortran 77 v8.5 ..... £275

GRAPHICS
- Corel Draw v2.0 ..... £279
- Cricket Graph v1.4 for Windows £59
- dGE - Database Graphics v4.1 ..... £179
- Dr HALO IV Paint & Imaging Pak £59
- FontWINDOW v4.1 ..... £89 (PLEASE CHECK!)
- GRASP with ARTOOLS v4.0 ..... £159
- GX EFFECTS v2.0 ..... £119
- GX Graphics £119
- GX Series Developers Pak £469
- Halo Professional £199
- Hijaak v2.1 ..... £89
- Inset Plus Hijaak £119
- MetaWINDOW v3.7 ..... £179
- MetaWINDOW/PLUS v3.7 ..... £239
- PCX Programmer's Toolkit v4.02 ..... £149

GUIs
- graphics-MENU v4.0 Pro-Pak £329
- graphics-MENU v4.0 with Data Entry £169
- Menuet v2.0 ..... £179
- WNDX Graphical User Interface Devlop £279

HARDWARE
- 1Mb 80ns SIMM Memory Upgrade £49
- Cardinal 2296 V.42 Fax Modem £135
- Cardinal 2450 V.42 MNP Modem £105
- Cardinal 9600 MNP-5 V.42 Modem £285
- Cyrix 83D87-33 Math Coprocessor £149
- Intel 80387 DX Math Coprocessor £109
- Intel SIMM 72 Pin 1Mb Memory PS/2 ..... £65
- ProScan PS1000 64 Grayscale Scann £399
- QuickTel 2400 Internal Modem £55
- QuickTel 9600 Internal Send Receive £89
- Typist Plus Graphics Hand Scanner £325
- ViVa 2400 Pocket Modem £69
- ViVa 2400I-Internal Modem £59
- ViVa 2400IM Internal Modem with MNP5 ..... £79
- ViVa 2442I Int Modem 42BIS, MNP5 ..... £115
- ViVa 2442I Fax/Modem V 42BIS £139
- ViVa 24I Internal 2400 Modem £79
- Zoom Internal Modem HC2400R £65

LAN
- Above LAN £199
- cc: Mail Platform Pak v3.2 ..... £99
- cc: Mail Remote v3.2 ..... £159
- Distinct TCP/IP for Windows £265
- Distinct TCP/IP for Windows SDK £330
- Emerald EMQ (NLM) £579
- FTP PC/TCP Development Kit v2.05 ..... £329
- LAN Assist Plus v3.1 ..... £119
- LanShadow £399
- LANtastic 2Mbps Starter Kit v4.0 ..... £259
- LANtastic AE-2 Ethernet Starter Kit v £379
- NE-2000 Ethernet Kit £119
- Netroom v2.0 ..... £55
- Netware Lite £55
- Novell Netware v2.2 -5 user £380
- Novell Netware v3.11 - 10 user £999
- Novell XQL for OS/2 ..... £410
- Saber File Manager v2.0 ..... £149
- Saber LAN Administration Pack v1.1 ..... £259
- Saber Menu for Windows £189

MAC
- 4th Dimension v2.2 ..... £399
- Adobe Photoshop v2.0 with ATM £419
- Design CAD 2D/3D £269
- Digital Darkroom £199
- Freehand £299
- Mac Tools Deluxe v1.2 ..... £55
- MPW (Mac Programmer's Workbench) £359
- MPW Development Environment v3.2 ..... £129
- OmniPage v3.0 for Mac £325
- SADE Editor v1.3 ..... £129
- Smalltalk/V for Mac £85

MEMORY MANAGEMENT
- 386 Max v6.0 ..... £49
- Above Disk + v4.0 ..... £45
- Headroom £55
- Instant Replay Professional v2.5 ..... £349
- QEMM 386 v6.0 ..... £45

MODULA-2
- ModBase - single user £45
- Multiscope Modula-2 Compiler v3.4 ..... £55
- Multiscope Modula-2 Devlop. Sys v3.4 ..... £139
- TopSpeed Modula-2 for DOS £65
- TopSpeed Modula-2 for OS/2 ..... £65
- TopSpeed Modula-2 Master edition £235
- TopSpeed Modula-2 Standard edititon £125

OBJECT ORIENTED PROGRAMMING
- Actor v4.0 ..... £159
- Matrix Layout v2.13 ..... £169
- Object Professional for C++ £145
- Object Vision v2.0 ..... £69
- Object-MENU £175
- Object Graphics £95
- Smalltalk/V 286 v1.1 ..... £85
- Smalltalk/V Communications £25
- Smalltalk/V Goodies £25
- Smalltalk/V v3.0 ..... £55
- Smalltalk/V-Windows £209
- TopSpeed Rogue Wave Class Library £65
- TURBO VISION DEVLPT TOOLKIT v2.0 ..... £85
- Whitewater Resource Toolkit £95

OPERATING ENVIRONMENTS
- AIX Advanced Interactive Executive £435
- Apple Mac Operating System v7.01 ..... £99
- Architech Operating System Multi-User £215
- DESQview 386 v2.4 ..... £99
- DESQview v2.4 ..... £59
- DR DOS V6.0 ..... £49
- DR Multiuser DOS £340
- OS/2 Standard Operating System £99
- QNX Operating System v2.15 ..... £315
- SCO UNIX System V/386 op. ys v3.2.2 ..... £319
- VM/386 Multitasking £119

OS/2
- Advanced Revelation for OS/2 ..... £539
- Asymetrix Toolbook (OS/2) £229
- BRIEF for OS/2 ..... £135
- Corel Draw for OS/2 ..... £279
- InstallShield for OS/2 ..... £399
- MS OS/2 Presentation Mgr.Softset £75
- Novell Xtrieve Plus for OS/2 v4.01 ..... £305
- PC/Focus for OS/2 ..... £579
- PDC Prolog OS/2 ..... £399
- Smalltalk/V-PM £209
- Super Project Expert for OS/2 ..... £365

OTHER
- allCLEAR (Procedural Charts) £135
- AskSam v5.0 (Free Form Database) £165
- Bar Code Library v2.3 ..... £269
- ColoRIX VGA Paint v1.3 ..... £75
- Dan Bricklin's Demo II Program £155
- Dance of the Planets £125
- Dr Solomon's Anti Virus Kit £105
- Encore (Music software) £279
- FlexScan 1 (Bar Code) £199
- Guide 3 for PC £259
- Install (Installation software) £129
- InstallShield for Windows (Installation) £269
- Labels Unlimited v4.0 ..... £49
- Microsoft Bookshelf (CD-ROM) £785
- Microsoft Source Profiler £38
- MKS MAKE v2.5 ..... £89
- MKS RCS v5.1 (Revision Control Sys) £159
- MKS Toolkit v3.7 ..... £149
- PKwareData Compression Libraries £179
- PKzip for DOS or OS/2 ..... £29
- Sourcerer's Apprentice Professional £285
- Spinnaker Plus Windows £199
- Survey Pro v1.2 (Survey Forms) £145
- The Stork (Installation Software) £119

PASCAL
- ASYNCH PLUS v5.04 ..... £99
- Asynch Professional £85
- MPW Object Pascal v3.2 ..... £129
- Pascal TOOLS £69
- Turbo Analyst v6.01 ..... £79
- Turbo Pascal for Windows £95
- Turbo Pascal Professional v6.0 ..... £129
- Turbo Pascal v6.0 with Objects £65
- TURBO VISION DEVLPT. TOOLKIT v2.0 .. £79
- TurboWINDOW/Pascal £89

SCIENCE, ENGINEERING & MATH
- Advanced Math Application Pack £59
- ASG Structural £349
- Chemical Engineering Applications Pac £55
- ChiWriter Deluxe v4.0 ..... £359
- ChiWriter v4.0 ..... £185
- Civil Engineering I £55
- Derive v2.0 ..... £159
- GAUSSX £79
- Graftool £329
- Grapher £119
- Graphlib v6.0 ..... £149
- Greenleaf Financial MathLib £199
- HOOPS DOS Extended £469
- Hiest/Hformat Optimum £89
- HYDSYS £710
- InGraf v6.0 ..... £199
- Infotech Notebook £689
- Lines £99
- Matfor £89
- Matfor - 386 ..... £259
- Mathematica 386 v2.0 ..... £419
- Mathematica 386/7 v2.0 fo Intel chip £595
- Mathematica for Mac v2.0 ..... £419
- Mathpak 87 ..... £69
- MathType/Windows v1.1 ..... £129
- Microstat II £239
- MISS £55
- Numerical Methods £49
- PlotLib v5.0 ..... £149
- SCHEMA III (SC-1) £299
- SSP/PC Scientific Subroutine Library £199
- StatView II £229
- Surfer £259
- Zortech C++ v3.0 Science & Enginrg £499

TRANSLATORS
- BAS_C Commercial £535
- FOR_C with Binary Runtime £345
- FOR_C with Runtime Library C source £489
- Pascal to C Translator £105
- Turbo to C £779

UNIX
- 386/ix X11 Development System £425
- C-ISAM for SCO XENIX 386 ..... £235
- Cosort for UNIX £369
- Edix Editor for UNIX 386 ..... £275
- FOR_C for UNIX/386 ..... £625
- Informix 4GL Compiled for SCO UNIX £1,100
- Interactive System V 386 Rel. 3.2 ..... £279
- LPI COBOL Compiler £799
- LPI FORTRAN Compiler £549
- Norton Utilities for SCO UNIX £149
- Oracle Tools & Database Unix 1 user £1,225
- SCO UNIX System V 386 Develop. Sys £549
- VEDIT PLUS for UNIX/XENIX £159
- Xtree for UNIX £149

WINDOWS DEVELOPMENT
- AceFile for Windows £149
- Asymetrix Toolbook v1.5 ..... £229
- Borland Resource Workshop £35
- Mathematica v2.0 for Windows £689
- MS Windows Device Driver Development £250
- Turbo Pascal for Windows £95
- TURBO VISION DEVLPT. TOOLKIT v2.0 .... £79

USA SOFTWARE

7-11 KENSINGTON HIGH STREET • LONDON W8 5NP
SALES: 071-938 4800 • SWITCHBOARD: 071 938 2161 • FAX: 071 937 5776

We will match any advertised Price!

- Prices exclusive of V.A.T. & shipping
- All major credit cards accepted.
- 30 days invoicing for qualified organisations.
- Special discounts for large or corporate buyers
- Prices subject to change without notice.

Over 3,500 products carried. Call today for price list-disk.

CIRCLE NO. 566

SOFTWARE PROTECTION!

Can You Spot The Difference?

- ☑ High-level security keys
- ☑ Assembler-based, customer specific, encrypted interrogation routines
- ☑ Over 140 language interfaces available
- ☑ Compatibility - due to 9 years experience & 750,000+ keys sold
- ☑ Reliable on-going support
- ☑ Physically unique keys for each customer
- ☑ MS-DOS, OS/2, UNIX, XENIX, WINDOWS 2 & 3, "MACINTOSH"
- ☑ Transparent operation with most peripherals on market
- ☑ Free language updates
- ☑ Parallel, Serial, Mac ports

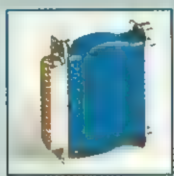
OVER 4,000 CUSTOMERS WORLDWIDE HAVE!

Electronic Key

The ideal device for identically produced software packages. Uniquely wired with customer code and a software code. Uses Assembler based program, decryption interface and random values.

1 Word Memory Key

Custom hardware wiring allows the developer total control over information stored in the key. 2 bytes of memory allows several packages to be protected with just the one key.



31 Word Memory Key

For multiple software protection schemes. 31 words of 16 bits of non-volatile dynamically programmable memory. Its capacity to store information provides virtually limitless power. Flexible protection scheme can be modified on-site during operation of software package.

UP3 Memory Key

Provides the ultimate in software security. Not tied to any language or O/S. 8 bit microprocessor powers from RS-232 level. Requires no power supply. For PC terminals, minis, & others using RS 232 C comms. Used on workstations. This key is effectively a computer.

Macintosh Key

Extremely powerful & customised protection for the Mac. 31 words available for random storage. Providing unequalled protection the Macintosh Memory Key connects to the SCSI 25 pin port and operates transparently.

MICROPHAR The European Leader*

* Based on the number of keys in use throughout Europe

Call us for more information or a demonstration package on

091-378 9191

CLEARSOFT,
Langley Moor, Durham DH7 8HG.
Telephone: (091) 378 9191 Fax: (091) 378 9393

Other European Countries

- **FRANCE,** Tel: 33-1-47-38-21-21
- **BELGIUM,** E2S (091 21 11 17)
- **GERMANY, AUSTRIA:** (06223 - 73730)
- **HUNGARY,** Polyware Kit (76-22-307)
- **ITALY,** Siosistemi (030 24 21 074)
- **PORTUGAL,** HCR (1 56 18 65)
- **SPAIN,** Hal 2000 (032 37 31 05)
- **SWITZERLAND,** SAFE (024 21 53 86)
- **THE NETHERLANDS,** E2S (015 15 88 37)

CIRCLE NO. 567

	Borland BGI	Microsoft	Topspeed	Watcom	Flash Graphics
Built-in Fonts	11	1	1	1	1
Compatibility	BGI	MS	MS	MS	FG
Fill Polygon	Y	Y	Y	Y	Y
Graphics Adapters	VGA/HERC/IBM8514a	VGA/HERC	VGA/HERC	VGA/HERC	VGA/HERC/IBM8514a
	ATT400/PC3270			SVGA	various SVGA cards
Loadable font format	BGI	.FON	-	-	Bitmaps only
Mouse support	N	N	Y	N	Y
Num of lib calls	82	86	46	93	57
Viewport Clipping	Y	Y	Y	Y	Y
World Coördinates	N	Y	N	Y	N

Please note that VGA refers to CGA/EGA/MCGA/VGA compatible graphic cards.

Figure 1 - List of Features

also be loaded, and again, some are available in the public domain. However, BGI doesn't support standard Windows (.FON) fonts.

MS Compatible Libs

The Microsoft library provides a similar interface to Borland's. Initialisation is quite straightforward, `_setvideomode(mode)` is all that is required - but it suffers from less flexibility than the Borland approach. There is built-in support for CGA, EGA, Hercules and VGA, but that's all. You can't easily add an SVGA driver.

Like Borland, Microsoft also has an invisible graphics cursor that provides an anchor for drawing lines and moving to another position on the screen. However, there is no equivalent call to Borland's `line()` function which takes both end-points. You can change the style of the line from Solid to a user-defined pattern, although it isn't possible to vary the width of the line.

All the shapes in this library are characterised by an extra parameter which determines whether the interior of the shape should be filled. This has the advantage of reducing the number of library calls, while still providing the most efficient method to fill a given shape. For instance the following code extract will draw and fill a rectangle.

```
_rectangle (
  _GFILLINTERIOR,
  x1, y1, x2, y2
        )
```

Microsoft differs significantly from Borland in the way that circular shapes are defined. When drawing an arc, you have to provide the end-points of two vectors that intersect the arc and this calculation requires a little floating-point arithmetic plus calls to `sin` and `cos` functions (see Figure 4).

Microsoft allows the programmer to set up both the origin and the size of a viewport that defines an area on the physical display which is used to output graphics. It is also possible to define a new world-coördinate system which maps an arbitrary coördinate system onto the physical device coördinates (ie 640 by 480 pixels for VGA). This enables you to use 'real world' measurements in your graphics applications without having to convert them to device coördinates yourself. These measurements are floating point and Microsoft provides floating point versions of its drawing functions to take advantage of world-coördinates (Topspeed doesn't support this feature so it lacks the floating point graphics functions).

There is only one built-in font, and that's the system font. However, Microsoft does provide you with a mechanism to load .FON files which means that you can use any of the Microsoft Windows fonts - so there's no shortage of exotic fonts.

	Borland	Microsoft	Topspeed	Watcom	Zortech
Clear Screen	1.6	3.9	3.8	2.4	4.5
Fill Polygon	1.8	1.2	2.2	1.2	8.1
FloodFill	5.6	2.6	2.9	2.5	124
Fill Block	1.5	1.8	3.5	1.5	2.7
Cycle Palette	8.5	17	17	17	0.05
Plot Pixel	15	27	21	28	9.3
Read Pixel	18	20	9.3	22	8.8
Draw Line	2.6	2.7	3.5	3.7	4.8
Draw relative Line	1.8	1.7	2.3	2.9	2.5
Draw Rectangle	2.9	1.5	1.5	5.5	1.7
Draw Polygon	1.8	1.8	2.5	3.0	2.6
Draw Circle	6.0	2.3	4.7	3.2	6.1
Draw Ellipse	8.3	3.9	6.4	4.4	8.3
Draw Sector	2.3	1.3	2.8	2.8	-
Draw Arc	6.04	3.1	7 6	8.2	2.8
Write Text	2.2	5.1	3.5	5.2	3.1
Xor Bitmap	1.8	1.7	1.9	1.2	-
Move Image	4.5	5.9	15	4.5	-
Copy Bitmap	1.8	1.6	1.8	1.1	6.8
Animation Demo	6.7	6.7	6.4	4.7	-
.EXE Graphics Index	3.7	3.4	4.4	4.0	4.2

- Cannot be implemented directly
All tests were measured in seconds using the ANSI `clock()` function.
The Graphics Index is an indication of the overall performance for each library - a lower index implies a faster library.

Figure 2 - Graphics Libraries Benchmark

Zortech Flash Graphics

I found the Flash Graphics API less intuitive than either the Borland or the Microsoft offerings. There is no graphics cursor so all shapes require parameters for both the start and the end-point coördinates (ie there is no equivalent to the `moveto` library call). When you draw a line it is necessary to specify both end-points. The problem is that Flash Graphics uses an array to hold these parameters, so before you start any drawing, you have to set up such an array of coördinates and pass its address to the shape drawing function.

Although the library cannot cope with user-supplied graphics drivers, Flash Graphics directly supports several exotic graphics cards as well as the normal ones, including the Orchid Prodesigner, Paradise VGA Plus and Trident VGA cards. The `fg_init()` library call automatically obtains the highest resolution for your graphics display.

Unlike BGI or Microsoft, there is no way to set common attributes such as colour or line

style. These parameters must be set up for each library call that produces graphical output (eg drawing lines or rectangles). There is also the question of the drawing mode. The two previous libraries provided a mechanism which allowed the programmer to change the way in which pixels were plotted (ie XORed with the background or SET) for all subsequent graphics calls. Since there is no common pool of attributes, in Flash Graphics this must be set for each library call. A Mask parameter, determining the colour planes onto which pixels are plotted is also required. Almost all of these parameters are needed to draw even the simplest of shapes such as a line, resulting in longer, more complicated function calls compared to the two previous libraries.

```
void fg_drawline
(
  fg_color_t colour,
  int mode, int mask,
  int line_type,
  fg_line_t line
)
```

As each shape in the Flash Graphics library contains enough information to redraw itself, it is easy to see how an object-oriented version of the library could be developed by encapsulating the whole library into a class library. This is exactly what Zortech has done with its Fg class library which also comes with the compiler and provides a C++ interface to the Flash Graphics engine.

Flash Graphics uses the same concept as BGI when drawing circular shapes - you only have to provide the radius and start/end angles (for Arcs). Unlike Borland, these angles are in the range 0 to 3600 which gives an accuracy of one-tenth of a degree (in Borland angles can only be specified in whole degrees).

As is apparent from the benchmark results of Figure 3, Flash Graphics doesn't have any library calls that draw sectors or manipulate bitmaps. This means that it is necessary to hand-code these functions ie it is not as complete as either the Borland or the Microsoft libraries. However, unlike Microsoft, Flash Graphics does offer a degree of SVGA support.

Graphic Library Benchmark overview

The benchmark consists of 20 distinct tests which were run on a 40 MHz 386 with an Orchid Prodesigner II SVGA graphics card and no floating point co-processor. Each benchmark used the 640 by 480 16 colour VGA mode and was compiled for the small memory model with optimisation switched off. Where necessary, the benchmarks were repeated in a loop to obtain measurable timings.

All of the libraries provide a function to clear the graphics screen. *Clear Screen* gives an indication of how long this takes. The next group of benchmarks look at three ways to fill shapes. *FloodFill* and *Fill Block* are used to fill a rectangular area of the screen. *Polygon Fill* fills the interior of a triangle. *Cycle Palette* which increments the red, green and blue elements of the background colour. *Plot Pixel* writes a dot to each screen coördinate and *Read Pixel* reads them back. The next eight tests cover the built-in shapes (ie lines, vectors, rectangles, polygons, circles, ellipses, arcs and sectors). A fundamental requirement of any graphics library is the ability to place text on the graphics screen. Using the BIOS system font, *Write Text* places 16 lines of text on the screen. Finally there are the image handling tests. *Xor Image* and *Copy Image* copy a large rectangular block of the screen to another location using the logical Xor and Copy (ie replace) mode respectively. *Move Image* is a simplistic animation sequence in which a small rectangular bitmap flows smoothly down the screen. The finale is the *Animation Demo* which is a more demanding animation that uses several graphics library calls.

The Graphics Index in the last row of the benchmark table summarises all the results. This value was calculated by taking the geometric mean of the results for each of the five libraries. If interested in the test source code, please call the .EXE office.

Benchmark Results

There are a few points worth bearing in mind regarding the benchmark results in Figure 2. First, let me draw your attention to the speed of the Flash Graphics 'Cycle Palette' test. How does it manage to cycle

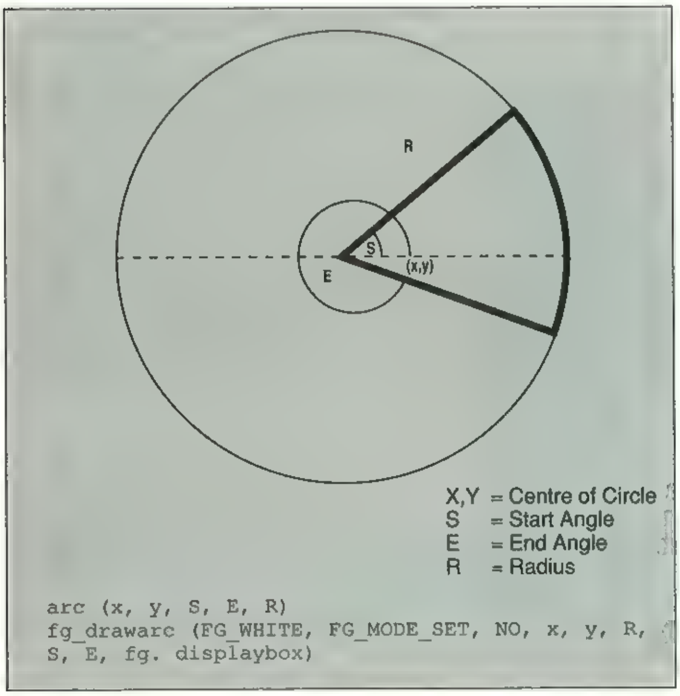


Figure 3 - Arc drawing with BGI/Flash Graphics

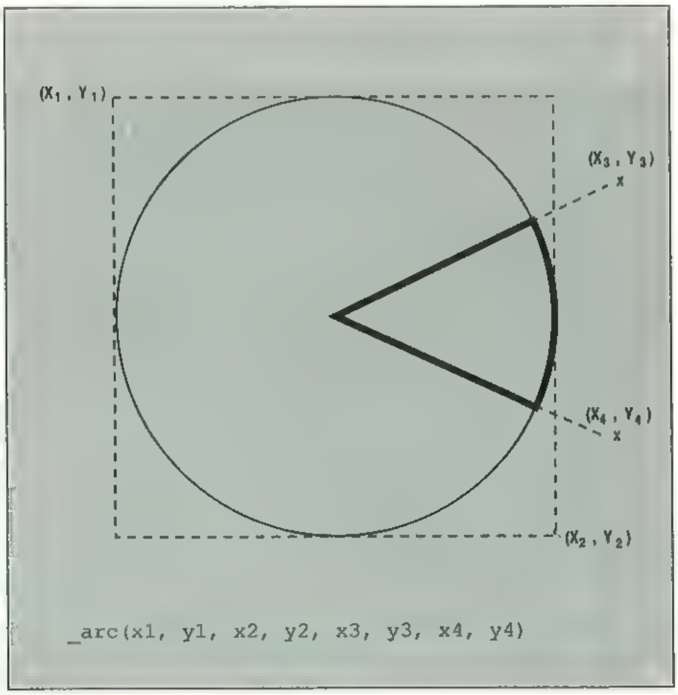


Figure 4 - Arc drawing with Microsoft Graphics library

the palette in 0.05 seconds, when the others take between 8.5 and 17 seconds to perform **exactly** the same task? Unlike the other libraries which use `int 10h` (the ROM BIOS video driver) to talk to the graphics card, Flash graphics writes directly to the hardware. This is possible because Flash Graphics provides specific hardware-dependent graphics routines for each of the graphics cards that it supports.

Only BGI and Flash Graphics provide calls which let you specify both end-points when drawing a line. I achieved the same result with the other libraries by using a combination of `movetos` and `linetos`. This additional overhead is reflected in the results for 'Draw Line'.

There are usually several ways to draw a complete circle if there isn't a library call that already does it for you (eg Borland's `circle` function). A circle is a special case of an ellipse in which the radius along the X-axis is equal to the radius along the Y-axis. I used this definition to produce the Microsoft, Topspeed, Watcom and Zortech circle drawing tests (ie with these libraries, the `ellipse` library call was used to output a circle).

Looking at the results for 'Draw sector' we can see that the Microsoft library is the clear winner in this test. What's interesting here is the fact that the two end-points of the sector had to be calculated using floating point arithmetic before being cast to integer values. Although BGI allows the programmer to specify these parameters as integer Start and End angles, the overall result is somewhat slower than Microsoft.

Conclusion

All the graphics libraries that I have tested were adequate for their purpose. However, there were significant differences between the offerings, which merit being taken into account when selecting a C compiler for DOS graphics development.

JPI's TopSpeed library probably offered the least, even though it attempted compatibility with the two graphics standards. In particular, its MS implementation was incomplete and slow. Solid old Watcom's graphics library is, well, a bit too solid - one would expect a bit more speed from the famous Canadian optimiser. Watcom carries off the consolation prize of 'best documentation', though. Symantech/Zortech's effort was perhaps the most bizarre. Points gained for the wide range of graphics adapters supported and the speedy low-level calls were rapidly lost by a clumsy API and some high-level primitives that were either missing or sloppily implemented.

This leaves Microsoft and Borland. According to my tests, the MS library emerged as the speediest of the bunch. Its API was very complete. We particularly admired the ability to use Windows fonts, and the facility to map world coördinates to screen coördinates. Nonetheless, Microsoft offered no help in supporting non-standard adpaters. So, for its reasonable performance, its intuitive API, its loadable device drivers and its excellent documentation, I vote in Borland's BGI as the best all-rounder in the bunch.

EXE

The following prices were supplied by The Software Construction Company and include p&p but not VAT. Borland C++ V3.0 - £189. Microsoft C V6.0 - £215. Topspeed C V3.0 (standard) - £109. Watcom C V8.5 - £319. Zortech C++ V3.0 (standard) - £249.

Complete Control

Paul Kemp has been jazzing up his Windows apps with some cool new controls from Blaise's Windows Control Palette.

Bored with Windows' standard controls? The Windows Control Palette (WCP) from Blaise Computing takes advantage of Windows DLLs to give programmers an extended collection of posh-looking custom controls. Because their default behaviour is predefined they are simple to use and require little programming effort.

Basics

At the heart of WCP is a DLL (CPALETTE.DLL) which, when loaded, registers the WCP controls so that they can be used like any other Windows control. This DLL can be called from any language that supports DLL access. Header files, example programs and static link libraries are supplied for Microsoft C (or compatible), Borland C++ (versions 2.0 and 3.0), Turbo C++ for Windows and Turbo Pascal for Windows (TPW) - full Visual Basic support will be available in March and will be supplied as a free upgrade to existing users of WCP. Support DLLs are also provided so that the WCP custom controls can be 'seen' by resource editors: Borland's Resource Workshop and Microsoft's Dialog Editor (with VB support in the upgrade). I particularly liked this feature, especially with Resource Workshop where the new controls rather neatly tack themselves onto the toolbar along side Borland's own custom controls and the standard Microsoft ones.

Controls in detail

The nine WCP controls are listed in Figure 2 and a selection are shown in Figure 1. The `CPDialog` class implements special 'fast painting' of WCP controls and also displays a textured background similar to Borland's 'chiselled steel' effect. `CPButton`, `CPRadioButton` and `CPCheckBox` controls are fully configurable so that the programmer can supply an appropriate set of bitmaps which represent the control in each of its possible states. There are attractive defaults provided and a great many more bitmaps available in another DLL (CPBITMAP.DLL) if you don't want to create your own. Once you have primed the control with the necessary bitmap handles, they are automatically switched at run-time in response to user interaction.

`CPStatic` controls are used to display static text, icons and bitmaps in dialogs or client window areas. Extra style settings are available for text display which give a raised or lowered appearance to the characters. In a similar style to Borland's `BorShade` control, WCP's `CPCanvas` class can be used to display a raised or lowered, shaded rectangle upon which other controls can be painted. Horizontal and vertical lines can also be drawn. This control would normally be used to group controls on a dialog or to separate portions of a window.

The most innovative of the new controls are `CPMeter`, `CPSpin` and `CPToolBar`. Unlike the other WCP controls which are enhanced versions of standard controls, they constitute a small set of entirely new interface elements. A `CPMeter` is one of those thermometer-style thingies that Windows install programs are so fond of (and so often abuse - have you noticed how they get to 100% and then rather cheekily go back to 0% and start all over again?). The application sets the range and progress of the creeping ooze, while the control calculates and displays a 'percentage complete'. The dimensions and colour combinations of the bar are also configurable.

`CPSpins` are spin button controls that consist of a pair of bitmapped buttons indicating *increase* and *decrease* (or *up* and *down* if you fancy). As with the other button-type controls, the application can supply bitmaps for these buttons (two for each of the pair, four in all) or leave the control to use its own defaults. Notification messages are sent to the control's parent window when the user clicks on either of the buttons.

My favourite of the bunch is the `CPToolBar`. A toolbar in WCP-speak is one of those free-floating chocolate bars as seen in Visual Basic and now common in many other Windows apps. After creating a tool-

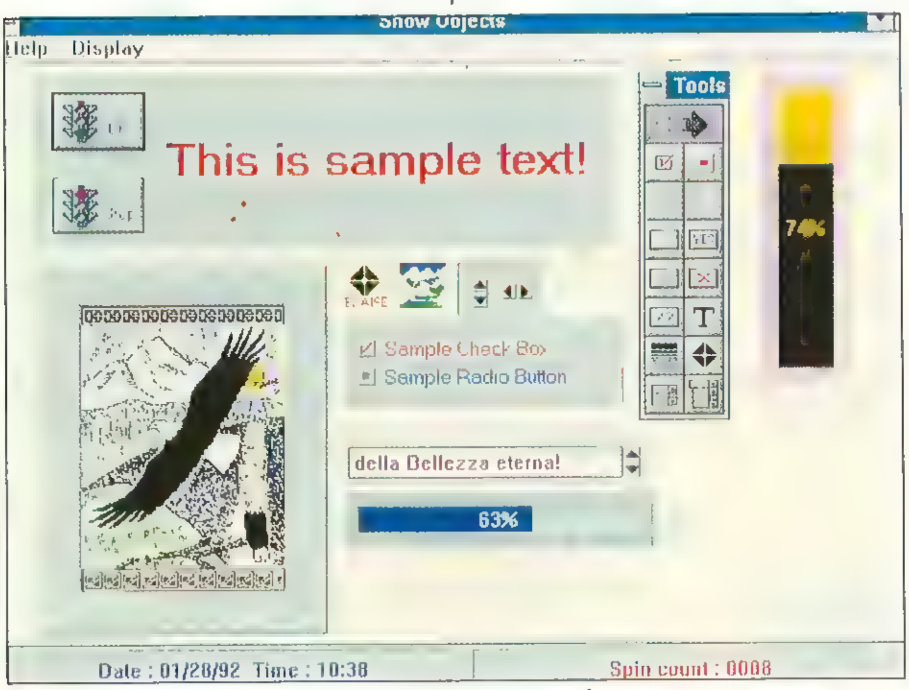


Figure 1 - WCP examples

ADVANCED BUSINESS GRAPHICS

WCP control name	OWL classes (C++ & TPW)
CPDialog	TCPDialog
CPButton	TCPButton
CPRadioButton	TCPRadioButton
CPCheckBox	TCPCheckBox
CPStatic	TCPStatic, TCPStaticBitmap, TCPStaticIcon, TCPStatusField
CPCanvas	TCPCanvas, TCPGroupBox, TCPSpinList, TCPStatusLine
CPMeter	TCPMeter
CPSpin	TCPSpin
CPToolBar	TCPToolBar

Figure 2 - WCP controls

bar object in the application, the programmer initialises a two dimensional array of bitmap handles which represent the component buttons in their 'up' and 'down' states. From here on in the bitmaps are automatically alternated in response to the user's mouse clicks, and messages are sent to the toolbar's parent indicating which button has been selected.

Programmer's perspective

In addition to supporting Blaise's own Windows C++ application framework Win++, WCP comes with a complete interface for use with Borland's ObjectWindows library (OWL). Figure 2 shows the relationship between WCP controls and OWL classes. The WCP classes for both TPW and BC++ are derived from OWL base classes. Two of these classes in particular extend the functionality of the basic WCP controls by encapsulating more complex behaviour.

The `TCPSpinList` class is a combination of three interface objects: a spin button, a static text control and a background canvas. It's a bit like a drop-down combo box without the drop-down. The object provides two built-in ways to control the values displayed in the text field. Using one way, you provide an array of character strings to display as the internal index (or *spin counter*) changes in value. In the other way, the actual numeric value of the spin counter is shown.

A `TCPStatusLine` is a special type of `TCPCanvas` which is associated with one or more `StatusField` objects (which are derived from `TCPStatic`). Status lines can be placed anywhere within the client area and have a member function to move them and the status fields in concert when the client area changes in size.

Conclusion

I liked the Windows Control Palette although I would have been happier with more detailed documentation and better example programs. It is a good example of a third party library stepping in to plug the gaps left in more mainstream development systems, even though there is some overlap with Borland's custom controls. I was particularly impressed by its comprehensive support for different languages and tools, and the ease with which the controls could be used. This is what code reuse is all about, lets hope we see more high quality add-ons like this in the future.

EXE

The copy of Windows Control Palette from Blaise Computing Inc was supplied for review by Grey Matter (0364 53499) who quoted a UK price of £99 excl VAT.

Get your act together!

Jules has been speaking to computer salesmen, this month. He's a bit distraught.

I have been a fan of the *idea* of portable computers for years - indeed I had one of the first palmtops about ten years ago (before there was such a word). Until recently, though, I was not so sure about the reality - most desktop machines are far more practical for most people than even the most fashionable portable. However, I think laptops are reaching the stage now where they are becoming genuinely useful, with sensible battery lives and reasonable compute power, so when a job came up that needed a laptop, I accepted it and went shopping. Most laptops and portables are (like most computers everywhere) DOS-compatible. I can't see the point of this - trying to type program names or (worse!) Wordstar keystrokes onto a tiny keyboard in a bucking car is not my idea of fun. No, DOS has its places, but planes, trains, and automobiles are not among them.

The problem was, having ruled out a DOS box, I was in uncharted territory! I found myself in the realms of operating systems, peripherals, and application programs that neither I, nor any of my colleagues, knew anything about. That is, I found myself in the same position that the vast majority of prospective computer users do every time they put their hand in their pocket! What does one do in such a situation? What I did was to buy several of the magazines which contain more advertising than editorial (you know the ones you take home in a wheelbarrow?), and looked through them. I found the respectable-looking suppliers, and I found the respectable-looking machines. I read a few reviews, and I eventually decided on a Holger 245 (not its real name, by the way!) complete with an accessory pack which made it actually useful. I contacted a firm who had a large colour photograph of the Holger in its advertising, and was quoting a pretty good price, and I rang up to place my order. 'Oh no', they said, 'We're not selling that any more. You see, there's a new version coming out next month, and we're going to sell those instead. Can I take your order for that?' Did they have any brochures for it? Did they know how much it was going to cost? Did they know when they were due to get some? Did they know who was going to make it? Nope! They knew nothing whatever, yet they still were prepared to take my order.

I was given no less than five different (and untrue) reasons why the machine had been deleted

I called the number of the manufacturer which they gave me, and found it had been disconnected. I eventually tracked the company down (they had moved - to a different country), and asked them what the story was. Yes, there was a new version, due out in about eighteen months. They were not planning to discontinue the old one. Could they sell me one? Yes, at twice the price the supplier had been advertising, and no, they wouldn't bargain. I called the supplier back, and they told me I had been misinformed. They *had* deleted the computer from their list, but for a wholly different reason to the one given earlier. To cut a long story short, over five separate phone calls I was given no less than five different (and incompatible, and untrue) reasons why the machine had been deleted. Good work, guys!

Over the next few days, I called a number of suppliers, and found a variation in price for identical equipment of 300%. The bag of accessories had been discontinued, and only one company (from perhaps 30) could tell me what configuration I actually needed. The Americans gratuitously wished me nice days, the British were characteristically sullen, and even the Asians wouldn't haggle. Eventually, I did something I swore I would never do, and bought second-hand. I got the equipment I wanted, I got enough help from the seller to get it going, I got some useful hints, and I got the price I wanted. I have disguised the names of the companies involved, not to protect their (or my) interests (you know who you are!), but because my experience is not unusual. I tried a similar exercise with other machines and with other companies and found the same thing.

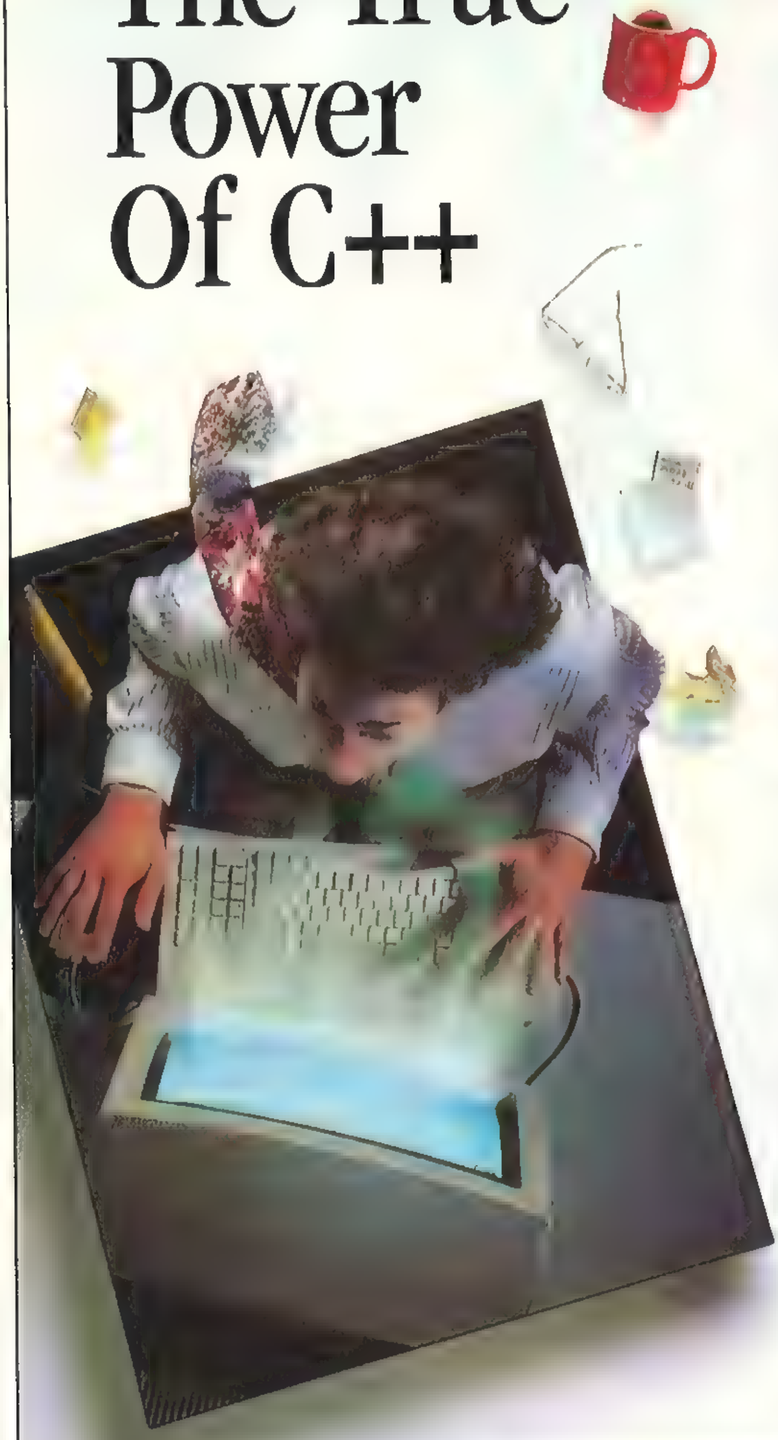
Of course, people joke about computer salesmen the way they joke about estate agents, but I have been so close to DOS boxes that I hadn't noticed - I always knew what I wanted, and didn't need to have it explained to me. What occurs to me, though, is that for a small-volume item with (apparently) hefty mark-up, the situation is pretty bad, but for DOS boxes, which are high-volume and very low mark-up indeed these days, the situation must be even worse. I thought only incurable technophobes said 'computers are too complicated', but I now see that, for the average user, such an experience would be so daunting he may well walk away from the problem. Is it my job (as a programmer) to sell the idea of a computer to a customer, or a secretary's job to sell the idea to her boss? What do these salesmen think they are being paid for? We hear a lot of complaint these days that the computer market is not what it was, and that we should feel sorry for the box-shifters because they can no longer charge 30% or 40% mark-up in a market which is reaching saturation. Cobblers! If the market is changing, then selling styles must change, and if that means telling customers what they need to hear in order to feel good about a purchase, so be it. Salesmen in every other industry know this, why not in ours? Come on, salesmen - get your act together!

EXE

Jules May is a keen user of both desk-bound PCs and at least one laptop. He doesn't sell computers, but is considering the possibility. He can be contacted on 0707 44185 or on cix as jules.


"...I had one of the first palmtops about 10 years ago..."

Unleash The True Power Of C++



It Takes More Than The Language.

Unleashing the full potential of C + + takes powerful object-oriented tools and a rich library of classes. Using only a C + + compiler and traditional tools yields traditional results – slipped schedules and maintenance nightmares. Only Objectworks\C + +, Release 2.4, provides the comprehensive development environment that allows you to produce extraordinary results.

Speed Your Development and Maintenance.

Objectworks\C + + provides dynamic, graphical browsers to illuminate the tangle of class relationships, object interactions, and program structure, as no static or textual information possibly can. The unique integration of the C + + source level debugger and interactive browsers ensure that essential information is instantly at your finger tips. And, a broad range of cross references are as close as a point and click. All of these tools aid in the understanding of code, increase reuse, decrease confusion, promote consistency and, ultimately, speed your development and maintenance.

An Open Environment.

Based in the latest AT&T C + + language System, Release 2.1, Objectworks\C + + provides support for SunView and X Windows on Sun platforms. Its open environment also allows you to use your favourite C preprocessor, C compiler, linker, profiler, or source code control system. The sophisticated class libraries included in Objectkit\C + + exploit the natural synergy between the development environment and the extensive reusable class libraries. Objectworks\C + + also allows programmers to use existing makefiles without any modification, saving time and money.

Team Programming.

Objectworks\C + + is the only C + + development environment that allows engineers to work cooperatively, loading and browsing one another's code, without interrupting each other. Programmers simply access an information file containing a description of the code they need, and then they query this description.

A I International are uniquely qualified to help you exploit the benefits of object-oriented technology and C + + programming applicable to your development efforts. If you'd like our technical management brief "Exploiting the Full Benefits of C + + Objects", or are ready to unleash the true power of C + +, call us today at 0442 876448

Without Objectworks, you're without objects.

A I INTERNATIONAL LIMITED

The Chapel, Park View House, 1 Park View Road, Berkhamsted, Herts, HP4 3EY
Telephone: 0442 876448 Fax: 0442 877997

CIRCLE NO. 603

Writing reuseable classes

Five Top Tips on C++ class design from Laine Stump, our man in Ankara.

After becoming completely swallowed by a new methodology, you sometimes make the mistake of thinking the rest of the world is in just as deep as you. Since I have spent the last three years madly buying books about C++ and experimenting with OOP, I of course assumed that all other programmers had similarly thrown out Kernighan and Ritchie in favour of Stroustrup, and Wirth in favour of Meyer. I was even worried that, in my absence from the 'software scene' in the US, I was falling behind the times. Somehow I had a vision that all the software houses were making a mass shift to Object Orientation, which would eventually render me obsolete.

But during a visit to Seattle this summer, we had dinner with a hardware engineer friend of mine. Eventually the discussion got around to C++ and OOP, and I was a bit shocked when he told me that, in August of 1991, his company was still using standard C for all their development. They were interested in OOP, he said, but were so pushed by management to produce results that none of the software people had been able to investigate. They hadn't even upgraded their Turbo C V2.0 compiler!

This situation seems to be quite widespread. Everyone knows 'OOP is good', but the corporate bean counters don't give the engineers enough time to learn how to take advantage of it. Many who have traded up to a C++ compiler are just using it as a glorified C compiler. It's really sad to see so many megabytes of disk space going to waste (a full installation of the latest Borland C++ takes over 40 MB!).

Thinking back through my own early experiences with OOP and C++ (which are still continuing!), I have come up with a few pointers to help those with limited time to avoid the rocks and whirlpools when they dive into the object-oriented river. The discussion of class design and reuseability that follows is not based on academic research, it is simply what I have found to work, and is open to criticism (please leave the rotten vegetables at home, though).

```
class Light {..};
class RedLight : public Light {..};
class YellowLight : public Light {..};
class GreenLight : public Light {..};
class TrafficLight : public RedLight,
  public YellowLight, public GreenLight {..};
```

Figure 1 - Excessive inheritance

Unless you have a good reason, prefer Wide classes over Deep classes

Reuse

Any good book will tell you that three of the most important features of OOP languages are Encapsulation, Inheritance, and Polymorphism. All three lead to one of the 'promises of OOP': Reuse. Most of us immediately embrace the first two features (which we have been secretly using in conventional languages anyway), but are a bit confused about the third. Unfortunately, inheritance and encapsulation are both severely crippled without proper doses of polymorphism (on the other hand, polymorphism is impossible without inheritance).

Polymorphism is realised in C++ via virtual functions. Basically, each instance of a class contains a table of pointers to the functions which have been designated as `virtual` for that class. At run time, rather than calling a function directly, it is called indirectly through the virtual function table (sometimes called virtual method table, or 'VMT'). The result is that the routine actually executed depends on the type of object for which it was called.

The Inheritance Tree

For some reason, many neophyte OOP programmers are so struck by the concept of inheritance that they want to inherit everything. This leads to class structures like that of Figure 1, an admitted extreme example. The obvious problem here is that we have tried to embed an attribute of an object (its colour) into the class definition. This not only limits the usefulness of the class, but makes it difficult to derive any new classes.

My 'First Hint to Reuseability': Don't embed simple attributes into the 'soul' of the class or make it more specific than necessary.

Wide vs Deep

Another symptom of the Inheritance Craze is what I call Deep classes. The `ColourGrDev` class in Figure 2 is an example of a Deep class. Aside from making it difficult to figure out what functions are available (you have to search back through the entire inheritance tree), this deepness causes problems within the class itself. For example, if

```
// Deep version
struct Dev
{
public:
  void open();
  void close();
};

struct GrDev : public Dev
{
  int cx, cy;
  void moveto(int x, int y);
  void lineto(int x, int y);
  void circle(int x, int y, int r);
};

struct ColourGrDev : public GrDev
{
  int  colour;
  void setcolour(int c);
};

// Wide version
struct ColourGrDev
{
  int cx, cy, colour;
  void open();
  void close();
  void moveto(int x, int y);
  void lineto(int x, int y);
  void circle(int x, int y, int r);
  void setcolour(int c);
};
```

Figure 2 - Deep vs Wide classes

```
class GradeList
{
  int *data;
  int count;
public:
       GradeList(int ct)
       {
         count = ct;
         data = new int[count];
       }
       ~GradeList()    { delete data; }
  int  Count()      { return count; }
  int  Get(int i) { return data[i]; }
  void Put(int i, int g) { data[i] = g; }
  int  Average()
  {
    int sum = 0;
    for (int ct = 0; ct < count; ct++)
      sum += data[ct];
    return sum/count;
  }
};
```

Figure 3 - GradeList #1 - poor encapsulation

you want `circle()` to be able to draw a coloured circle it must have access to the current colour, but `circle()` is defined in `GrDev`, while the colour is not available until `ColourGrDev`. You could redefine the members of `GrDev` in `ColourGrDev`, but that would render `GrDev` itself nearly worthless, as all its members except `moveto()` need colour, so they would all be redefined.

Unless the intermediate classes serve some other purpose (for example, maybe other types of devices will have `open()` and `close()`, but no line drawing), I prefer to use Wide classes. The second definition of `ColourGrDev` in Figure 2 is Wide. The advantages of Wide classes are that they are easier for client programmers to understand, and they eliminate the member visibility confusion we saw above.

The 'Second Hint to Reuseability'? I may catch some shrapnel on this one, but: Unless you have a good reason, prefer Wide classes over Deep classes. If you have more than two levels of inheritance before you get to a class that is instantiated (an object is declared), you should be concerned.

```
class GradeList
{
  int *data;
  int count;
public:
                GradeList(int ct)
  {
    count = ct;
    data = new int[count];
  }
  virtual       ~GradeList()
                { delete data; }
  virtual int   Count()
                { return count; }
  virtual int   Get(int i)
                { return data[i]; }
  virtual void  Put(int i, int g)
                { data[i] = g; }
  int           Average()
  {
    int sum = 0;
    for (int ct = 0; ct < Count(); ct++)
      sum += Get(ct);
    return sum/Count();
  }
};
```

Figure 4 - GradeList #2 - better encapsulation

Polymorphism

A related mistake is limiting the 'virtual functionality' of a base class. Imagine a `Window` class which is used to derive a `ScrollingWindow` class. Usually the windows of a program are kept in some kind of a list, which in this case would be a list of pointers to `Window` (a `Window*` can point to a `ScrollingWindow`, but the inverse is illegal). We access member functions of these windows through the `Window*`, but since a `Window*` only allows us to call the members which were defined in `Window`, we can't call the functions associated with scrolling.

One solution in this case is to declare empty virtual functions for scrolling in `Window`, and redefine them in `ScrollingWindow`. Now you can call the scrolling functions with a `Window*`, and through the magic of virtual functions, if the window is a `Window` nothing will happen, while if it is a `ScrollingWindow` the appropriate action will be taken.

And so we arrive at the 'Third Hint to Reuseability': Declare as much functionality as possible in the base class using virtual functions; redefine those functions in descendant classes. If this functionality seems out of place in the base class, maybe you need to rename your base class.

Encapsulation

Traditionally, encapsulation has been closely linked to data hiding. The idea is to capture all data structure-dependent code within the functions of the class, resulting in an independence of the class and its client programs. For example, it makes no difference to clients of the `GradeList` class in Figure 3 whether its data is stored as an array, or as a linked list; all client accesses to the data are through `GradeList`'s member functions. This has the dual advantage of removing complexity from the client program as well as allowing different implementations of `GradeList` to be tried without changing the client program.

Remember, though, that users of a class won't just create instances of that class; they may also derive new classes from the original class. If you decide to create a `GradeListFile` which keeps the grades in a file on disk you won't want to rewrite everything. Instead you would like to inherit all the features of the original `GradeList`, changing only features (functions and data) which must be different from the original. That is, after all, the promise of Reuse.

But, although the algorithm for the `Average()` function will be the same for both classes, if you inherit from the `GradeList` of Figure 3, you'll have to rewrite `Average()`, as it accesses `GradeList`'s data members directly, while data in the new class will be stored on disk. The problem is that because the member functions of `GradeList` are tightly coupled to the data, they cannot be reused in a derived class where the data is different.

With a bit of rework to separate the data of the class from its functionality, encapsulating direct data references within *a portion of the class*, you can arrive at the base `GradeList` class in Figure 4, which defines `Average()` in terms of other functions of `GradeList`. Now when you derive `GradeListFile` you will still define `Get()`, `Put()` and `Count()`, but you won't need to redefine `Average()`, as it is defined completely in terms of `Get()` and `Count()`. By defining `Get()` and `Count()` as `virtual` in the base class, you assure that `Average()` will call the proper version for the current `GradeList` (or a descendant) object.

This leads us to the 'Fourth Hint to Reuseability': Encapsulate direct dependencies on the data structure of a class in as few, simple

```
class GradeList
{
public:
  virtual       ~GradeList()    { }
  virtual int   Count()     = 0;
  virtual int   Get(int i)  = 0;
  virtual void  Put(int i, int g) = 0;
  int           Average()
  {
    int sum = 0;
    for (int ct = 0; ct < Count(); ct++)
      sum += Get(ct);
    return sum/Count();
  }
};
class GradeListArray : public GradeList
{
  int *data;
  int count;
public:
  GradeListArray(int ct)
    { count = ct; data = new int[count]; }
  virtual ~GradeListArray()
    { delete data; }
  virtual int Count()
    { return count; }
  virtual int Get(int i)
    { return data[i]; }
  virtual void Put(int i, int g)
    { data[i] = g; }
  // inherits original Average()
};
class GradeListFile : public GradeList
{
  FILE *data;
public:
  GradeListFile(char *name, int ct)
    {// open file }
  virtual ~GradeList()
    { // close file }
  virtual int Count()
    { // read count from file }
  virtual int Get(int i)
    { // read element from file }
  virtual void Put(int i, int g)
    { // write element to file }
  // inherits original Average()
};
```

Figure 5 - GradeList #3 - abstract base class

functions as possible, defining other functions in terms of those dependent functions; declare the dependent functions as `virtual`. Derived classes will only need to redefine the virtual functions.

Sometimes the 'data' of a class is really represented by a member function. Take, for example, a `Piece` class with descendants `cRectangle` and `cEllipse` (see *Segments From Hell* .EXE Feb '92). A rectangle and an ellipse can be defined with the same data (the coördinates of the enclosing rectangle), but they each use that data in a slightly different way - calling the appropriate function from the graphics library. In this case, a virtual `Paint()` function should be defined in the base class, and all other functions (`MoveAbs()`, `MoveRel()` etc) should be defined in terms of the `Paint()` function.

Abstract Classes

There is still a problem with version two of `GradeListFile` - although it inherits the `data` and `count` members from its base class, it never uses them. This doesn't create any extra work for the programmer, but the two data members are wasting memory. Solving this problem requires an *abstract* base class. An abstract class is one which is never intended to have an instance, it should only be inherited from. In C++, any class which has a pure virtual function (a virtual function which is declared, but not defined) is considered to be abstract.

To arrive at the abstract base class in Figure 5, we simply make a base class which is the same as the base class of Figure 4, except that the virtual functions are declared as:

```
virtual xxx(...) = 0;
```

rather than giving a definition for the function ('= 0' means 'pure'). After removing the bodies of all virtual functions, we remove any data members which are not referenced in the non-virtual functions. This yields a base class which can be used to derive both a `GradeListArray` class and a `GradeListFile` class. Notice that we have still succeeded in writing `Average()` only once.

Finally, the 'Fifth Hint to Reuseability': if you think that you may derive other classes from a class, create an abstract base class by making all virtual functions pure, and removing data members which are not referenced directly by the non-virtual functions.

Further Investigation

Although I have used C++ in my examples, everything I have discussed is valid for any object-oriented language. If you're interesting in learning more about object-oriented design in general, and don't mind looking at a different language, an extremely good source is *Object-Oriented Software Construction* by Bertrand Meyer (Prentice Hall, 1988). This book illustrates discussions with the Eiffel language (which resembles a cross between Pascal and Ada). Although not useful for looking up C++ syntax (obviously), it has been invaluable in helping my mind shift into object-drive.

EXE

Laine Stump is currently suffering from back pains due to his habit of carrying a shoulder bag crammed with OOP books wherever he goes. The fact that his favourite computer book store is over 10,000 km from his house is suspected to be a major contributing factor to his ailment.

Laine can be reached via the PC Tech BBS at (0101-612-345-4656, evenings, US time) or by post at: Bilkent University, Lojmanlari 3/9, 06533 Bilkent / Ankara, TURKEY.

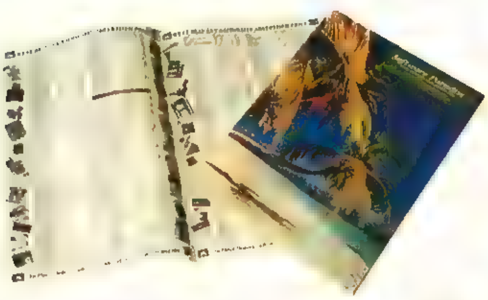
A WORLD OF

The widest range of software is only a phone call away

FAST - Federation Against Software Theft says £300m was lost by U.K software publishers alone through piracy in the year to March 1990. Software Paradise support and encourage the ethical use of software. To report copyright violations call FAST on (0628) 660377

The Widest Selection of Applications and Development Tools Anywhere

Place Orders Free on 0800-378-873



Just Call Us - It's That Easy!

Why struggle with traffic? Why fight crowds? Just pick up the phone and say hello to the convenient way to buy ALL your PC software. Software Paradise Expressline is the fast, new distribution system for all of today's popular titles. Choose from the widest range of software available from any single source, all listed in our FREE 100-page Buyers' Guide. Place your orders FREE and the software will be despatched to you directly.

Software Paradise has supplied 100,000's of customers since 1986 and has fast become the U.K's leading software supplier. Our service is both convenient and affordable with up to 50% off manufacturers RRP's and the widest range anywhere!

Call us today and discover a World of Software!

Software Paradise®

MORE SOFTWARE THAN ANYONE ELSE!

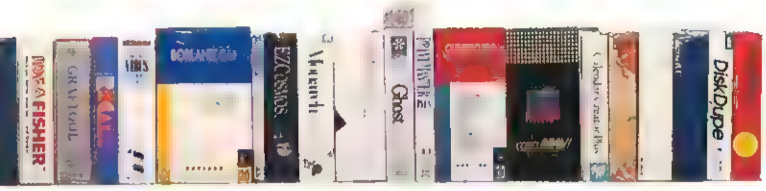
Call 0222 887521 • Fax 0222 862209

SOFTWARE *To Go!*

Over 15,000 Software

Products for

DOS, Windows™,



OS/2®, Mac

& UNIX®

* If you find an identical software product advertised in the national computer press by another mail-order company based in the U.K at a legitimate, current price - tell us at the time of placing your order and we will happily BEAT that price and SAVE YOU MONEY! Join the 1000 s of customers who have already benefited!

Don't forget - We'll BEAT Any Nationally Advertised Price! *

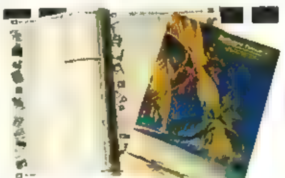
WHAT MAKES US DIFFERENT

- ✔ FREEPHONE 0800-378-873 Orders Only
- ✔ FREEPOST Mailing Address
- ✔ 24-Hour FAX Orderlines
- ✔ Up to 50% Discount On Over 15,000 Software Titles
- ✔ Pre-Sales Consultation
- ✔ FREE Post-Sales Technical Support
- ✔ Bulletin Board 0222 865551 (2400,8,N,1)
- ✔ Compuserve Mailbox 100015, 3570
- ✔ Unbeatable Prices *
- ✔ Next Day Delivery
- ✔ Huge Software Inventory
- ✔ FREE Literature Department
- ✔ Friendly & Knowledgeable Sales Staff
- ✔ VISA, ACCESS & Mastercard Accepted
- ✔ No Credit Card Charges Until Shipping
- ✔ Corporate P.O's Welcome
- ✔ Educational & Volume Discounts
- ✔ Latest Versions Guaranteed with Full Documentation & Manufacturers Warranties
- ✔ Over 6 Years Experience Providing the Very Best in U.K and U.S Software
- ✔ Stable & Secure Company with Impressive Track Record & Enviable Balance Sheet

Products shown in this advertisement represent only a small sample of our full range. Please specify Media Size when ordering. Goods are not offered on a trial basis. All our prices exclude Carriage & VAT.

Software Paradise, Avenue House, King Edward Avenue, Caerphilly, Mid Glamorgan, CF8 IHE

FREEPOST
-TODAY-
NO STAMP NEEDED



☐ Please rush me my **FREE** copy of *'The Discerning Persons Guide to Software'*

NAME

COMPANY

ADDRESS

TEL FAX

Mail to: Software Paradise, FREEPOST, Caerphilly CF8 1ZZ
.EXE 3/92

CIRCLE NO. 579

Efficient PC Serial Communications

Standard BIOS functions provide minimal support for serial communication. John Davies shows how some general purpose assembly routines can greatly improve matters.

Two previous articles in *.EXE* magazine by Andrew Margolis - *The Thinking Programmer's Guide to UARTs* (Dec 1989) and *If I May Interrupt* (March 1990) - have covered in some detail the operation of the PC UART. The latter article developed code for a PC-based terminal emulator which would run under interrupts. To make full use of this code one further step is needed which is to develop general purpose software so that serial interrupt handlers can be included in applications programs. The code should be callable from either assembly language or high-level languages such as C. This article takes the code for the terminal emulator and develops general purpose PC serial interrupt code.

Program Interfaces

The first decision to be made is what interfaces to provide for the applications program. The four obvious interfaces are: Start the serial interrupts, write a character to the output buffer to be transmitted, read a received character from the input buffer and stop the serial interrupts.

Two other interfaces which will be useful are routines to return the number of characters currently in the transmit buffer awaiting transmission and the number of characters in the receive buffer which have not yet been read. The reason why these interfaces are useful will become apparent in due course.

```
;***************************************
;    Serial port interrupt drivers
;***************************************
 NAME SERIAL PORT
 ASSUME CS: TEXT,DS: TEXT
  TEXT segment byte public 'CODE'
;***************************************
; Public procedures
; ------- ----------
 PUBLIC  start_serial_ints
 PUBLIC  stop_serial_ints
 PUBLIC  send_sio
 PUBLIC  read_sio
 PUBLIC  _no_rx_chars
 PUBLIC  _no_tx_chars
;***************************************
; Equates
; - ----
; Serial interrupt is no 0CH
 comint          EQU 0CH
; Base address of UART
 dataport        EQU 03F8H
; Interrupt enable register
 ierreg          EQU dataport+1
; Interrupt identity register
 iirreg          EQU dataport+2
; Line control register
 wordef          EQU dataport+3
; Modem control register
 modcont         EQU dataport+4
; Line status register
 statport        EQU dataport+5
; Modem status register
 modstat         EQU dataport+6
; Settings for RTS, DTR, OUT2
 set_lines       EQU 00001011B
; Settings for UART intrpt mask (rx,tx)
 int_setting     EQU 00000011B
; Settings to disable interrupts
 int_disable     EQU 00000000B
; Base address of PIC
 pic             EQU 20H
; PIC status register
 pic_stat        EQU pic+1
; End of interrupt command
 pic_eoi         EQU 20H
; Mask to enable interrupts in PIC
 int_mask        EQU 11101111B
; Size of receive buffer
 max_rx_buffer   EQU 1FFFH
; Size of transmit buffer
 max_tx_buffer   EQU 1FFFH
;***************************************
;      Start serial interrupts
;***************************************
start_serial_ints PROC FAR
; Save registers
 PUSH BP
 PUSH DS
 PUSH AX
 PUSH DX
 PUSH BX
 PUSH ES
; Get the required port setting
 MOV BP,SP
 MOV AX,[BP+16]
; Load DS with the relevant value
 PUSH CS
 POP DS
 CALL set_port        ; Set port parms
 CALL vectors         ; Set up IRQ and
 CALL intson          ; enable interrupts
; Restore registers
 POP ES
 POP BX
 POP DX
 POP AX
 POP DS
 POP BP
 RET
start_serial_ints ENDP
; Set serial port
;   -------- ---
set_port PROC NEAR
; Set up port to the parameters
; defined in AL
 MOV AH,0
 MOV DX,0
 INT 14H
 RET
set_port ENDP
; Getting old, setting new interrupts
; ------  ------  ---- ------  --
vectors PROC NEAR
 MOV AH,35H                 ; Get old irq
 MOV AL,comint
 INT 21H
 MOV oldint,BX
 MOV oldint+2,ES
 MOV DX,OFFSET service      ; Get new irq
 MOV AH,25H                 ; Set vector
 MOV al,comint
 INT 21H
 RET
vectors ENDP
; Activating interrupts
; --- - --------  ---
intson PROC NEAR
 XOR AX,AX                  ; Zero ptrs
 MOV nextin,AX
 MOV nextout,AX
 MOV charin,AX
 MOV charout,AX
 IN AL,pic_stat             ; Read PIC mask
 AND AL,int_mask
 OUT pic_stat,AL
 MOV DX,modcont             ; Set modem ctrl
 MOV AL,00001011B           ; DTR,RTS,OUT2 on
 MOV flowin,AL              ; Input flow ctrl
 OUT DX,AL
 MOV DX,modstat             ; Modem status
 IN AL,DX                   ; Read it
 AND AL,00010000B           ; Isolate CTS
 MOV flowout,AL             ; O-put flow ctrl
 MOV DX,ierreg              ; Int enable reg
 MOV AL,00001111B           ; Enable all
 OUT DX,AL
 STI                        ; Enable ints
 RET
intson ENDP
;***************************************
;      Stop serial interrupts
;***************************************
```

Figure 1 - Listing of serial port interrupt drivers (Continued overleaf)

```
_stop_serial_ints PROC FAR
; Save registers
  PUSH DS
  PUSH AX
  PUSH DX
; Load DS with the relevant value
  PUSH CS
  POP DS
  XOR AL,AL           ; Send 0 to int reg
  MOV DX,ierreg
  OUT DX,AL
  MOV DX,oldint
  MOV DS,oldint+2
  MOV AH,25H
  MOV AL,comint
  INT 21H             ; Restore old vec
; Restore registers
  POP DX
  POP AX
  POP DS
  RET
 stop_serial_ints ENDP
; Interrupt service routine
; ------------------------
service PROC NEAR
  PUSH DS
  PUSH AX
  PUSH BX
  PUSH DX
  PUSH SI
  PUSH CS
  POP DS                    ; Data is in
                            ; code space
again:
  MOV DX,iirreg             ; Int id reg
  IN AL,DX
  TEST AL,00000001B         ; int pending?
  JNZ endserv               ; no = quit
  MOV BX,OFFSET intab       ; jump table
  XOR AH,AH
  AND AL,00000111B          ; 16550 UART?
  ADD BX,AX
  CALL WORD PTR [BX]        ; Service int
  JMP again                 ; And loop back
endserv:
  MOV AL,pic_eoi            ; end of int
  OUT 20H,AL
  POP SI
  POP DX
  POP BX
  POP AX
  POP DS
  IRET
service ENDP
; Receive interrupt
; ----------------
rxint PROC NEAR
  MOV DX,dataport           ; Read UART
  IN AL,DX
  MOV SI,nextin
  MOV BX,OFFSET inbuf       ; Save data in
                            ; circ buffer
  MOV [BX+SI],AL
  INC nextin
  AND nextin,max_rx_buffer ; wrap around
rxpoll:
  MOV DX,statport           ; more data?
  IN AL,DX
  TEST AL,00000001B
  JNZ rxint                 ; yes=deal
                            ; with it
  TEST flowin,00000010B
  JZ end_rxpoll             ; If RTS is
                            ; already off
  SUB SI,nextout            ; bytes used?
  AND SI,max_rx_buffer      ; wraparound
  CMP SI,max_rx_buffer-100H ; nearly full?
  JB end_rxpoll             ; No
  MOV DX,modcont
  MOV AL,00001001B          ; RTS off
  MOV flowin,AL             ; Save RTS
  OUT DX,AL
end_rxpoll:
  RET
rxint ENDP
;***************************************
; Read character from input buffer
;***************************************
_read_sio PROC FAR
; Save registers
  PUSH DS
  PUSH SI
  PUSH BX
  PUSH DX
; Load DS with the relevant value
  PUSH CS
  POP DS
  XOR AX,AX
  MOV SI,nextout
  CMP SI,nextin             ; Buffer empty?
  JNZ read_buf              ; Yes=return
  MOV AX,100H               ; Data invalid
  JMP end_read
read_buf:
  MOV BX,OFFSET inbuf
  MOV AL,[BX+SI]            ; Read data
  INC SI
  AND SI,max_rx_buffer      ; Wrap around
  MOV nextout,SI
  TEST flowin,00000010B     ; If RTS on
  JNZ end_read              ; Then exit
  MOV BX,nextin             ; Else check
                            ; how much space
  SUB BX,SI
  AND BX,max_rx_buffer      ; Wrap around
  CMP BX,200H               ; Getting empty?
  JB end_read               ; No=exit
  MOV AH,AL                 ; Save data
  MOV DX,modcont            ; Modem ctrl
  MOV AL,00001011B          ; Turn RTS on
  MOV flowin,AL             ; Save RTS
  OUT DX,AL
  MOV AL,AH                 ; Restore data
  MOV AH,0                  ; Data valid
end_read:
; Restore registers
  POP DX
  POP BX
  POP SI
  POP DS
  RET
_read_sio ENDP
; Transmit interrupt
; -----------------
txint PROC NEAR
  CMP flowout,0             ; If CTS off
  JZ end_txint              ; Don't send
  MOV SI,charout
  CMP SI,charin             ; Check for
                            ; data available
  JZ end_txint
  MOV BX,OFFSET outbuf
  MOV DX,dataport
  MOV AL,[BX+SI]            ; Get data
  OUT DX,AL                 ; And send it
  INC charout
  AND charout,max_tx_buffer ; Wrap around
end_txint:
  RET
txint ENDP
;***************************************
;   Put character in output buffer
;***************************************
_send_sio PROC FAR
; Save registers
  PUSH BP
  PUSH SI
  PUSH DS
  PUSH BX
  PUSH DX
; Read character from stack
  MOV BP,SP
  MOV AL,BYTE PTR [BP+14]
; Load DS with the relevant value
  PUSH CS
  POP DS
  MOV AH,0FFH
  MOV SI,charin
  MOV BX,SI
  INC BX
  AND BX,max_tx_buffer      ; Wrap around
  CMP BX,charout            ; 1 space left?
  JNZ send_char
  MOV AX,0                  ; No space=ret 0
  JMP end_send
send_char:
  MOV BX,OFFSET outbuf
  MOV [BX+SI],AL            ; Save char
  INC charin
  AND charin,max_tx_buffer
  MOV AX,1                  ; Char sent=ret 1
  CMP SI,charout            ; Buffer empty?
  JNZ end_send              ; Not empty=end
  MOV DX,statport           ; Line status
  IN AL,DX
  TEST AL,00100000B         ; Transmit holding
                            ; register empty?
  JZ end_send               ; No=return
  CALL txint
end_send:
; Restore registers
  POP DX
  POP BX
  POP DS
  POP SI
  POP BP
  RET
_send_sio ENDP
; Modem status interrupt
; ----------------------
modint PROC NEAR
  MOV DX,modstat            ; Modem status
  IN AL,DX                  ; Read it
  TEST AL,00000001B         ; CTS changed?
  JZ end_modint             ; No=ignore
  AND AL,00010000B          ; Isolate CTS
  MOV flowout,AL            ; Save as flag
  JZ end_modint             ; Resume if on
  JMP txint
end_modint:
  RET
modint ENDP
; Line status interrupt
; ---------------------
exint PROC NEAR
  MOV DX,statport           ; Status port
  IN AL,DX                  ; Clear by
                            ; reading line
status
  RET
exint ENDP
;***************************************
; Calculate no of chars in output buffer
;***************************************
_no_tx_chars   PROC FAR
; Save register
  PUSH DS
; Load DS with the relevant value
  PUSH CS
  POP DS
; no of tx chars =
; (head ptr - tail ptr) AND max_tx_buffer
  MOV AX,charin
  SUB AX,charout
  AND AX,max_tx_buffer
; Restore register
  POP DS
  RET
_no_tx_chars   ENDP
;***************************************
; Calculate no of chars in input buffer
;***************************************
_no_rx_chars   PROC FAR
; Save register
  PUSH DS
; Load DS with the relevant value
  PUSH CS
  POP DS
; no of rx chars =
; (head ptr - tail ptr ) AND max_rx_buffer
  MOV AX,nextin
  SUB AX,nextout
  AND AX,max_rx_buffer
; Restore register
  POP DS
  RET
_no_rx_chars   ENDP
;***************************************
;     Data area for subroutines
;***************************************
  nextin  dw 1 dup (?) ; inbuf head ptr
  nextout dw 1 dup (?) ; inbuf tail ptr
  charin  dw 1 dup (?) ; outbuf head ptr
  charout dw 1 dup (?) ; outbuf tail ptr
  flowin  db 1 dup (?)
  flowout db 1 dup (?)
  inbuf   db max_rx_buffer dup (?)
  outbuf  db max_tx_buffer dup (?)
; Jump table for int service routine
  intab   dw modint        ; Modem status
          dw txint         ; Tx interrupt
          dw rxint         ; Rx interrupt
          dw exint         ; Ext / status
  oldint  dw 2 dup (?)     ; Old int vec
  _TEXT ENDS
```

Figure 1 - Listing of serial port interrupt drivers (Continued)

Program Notes

Before considering the code in detail a few points should be noted.

First, the Turbo C V2.0 subroutine calling conventions have been used throughout when passing parameters to or receiving parameters from subroutines. These conventions will also need to be observed when using these routines with assembly language. The Turbo C manual gives full details of these calling conventions but briefly parameters are passed into a subroutine on the stack and any value returned from a subroutine is returned in the AX register. The calling routine is also expected to clean up the stack after the subroutine returns.

Secondly the original terminal emulator code has been modified so that input and output buffers are now 8 KB each.

Thirdly all the routines accessible to the applications program have been declared as FAR routines. This is so that they can be used with all compiler models without modification. Declaring these routines as FAR adds a small but negligible overhead to the interrupt response time.

Finally the code applies only to the serial port COM1, the equates at the top of the assembly file would need to be changed to operate with COM2. Alternatively the code could be expanded to cope with both ports (simultaneously if necessary).

Let's now consider each of these routines in more detail. The descriptions of the code should be read in conjunction with Figure 1.

Serial Interrupts

Starting the serial interrupts is the same as the terminal emulator program except for two minor differences. All the registers which are used are saved on entry and restored on exit. This is necessary as this code may be called from anywhere in the applications software. Secondly, a parameter is passed to the routine which defines the required port setting. This parameter is a single byte which is defined as the same as the settings byte in the BIOS interrupt 14H, function 0. This restricts the setting of the port to those allowed by the BIOS but greatly simplifies the initialisation. The Turbo C function prototype for this function is:

```
void start_serial_ints( char )
```

Stopping the serial interrupts is exactly the same as the terminal emulator. The UART interrupt is disabled and the old vector restored. No parameters are passed to this routine or returned from the routine.

The Turbo C function prototype for this function is:

```
void stop_serial_ints( void )
```

Reading and writing

Writing involves passing a character to the `send_sio` routine which is put into the output buffer. If the buffer is empty then the character is written to the UART and the serial interrupts started. If the output buffer is full then the character is discarded. An integer parameter is returned from this routine indicating whether the character was successfully inserted into the buffer (1 = character sent, 0 = character not sent).

The routine to read a character from the input buffer is also very similar to the terminal emulator except for the way in which the data is returned. The terminal emulator used the carry flag to indicate whether there was valid data available or not. This is not really feasible for a high level language so an integer is returned instead. If the high byte of the integer is zero then valid data is available in the low byte of the integer. If the high byte is non-zero then no valid data is available. This could be expanded to return error codes in the high byte if, for example, a character was received with an error. The Turbo C function prototype for this function is

```
int read_sio( void )
```

Calculating characters

Calculation of the number of characters awaiting transmission (`no_tx_chars`) requires the use of the head and tail pointers of the transmit buffer. The returned value can be very useful if there is a need to write a large number of characters in one block to the buffer. It would be possible to write each character in turn and check the return value but it is much simpler to check the space available first before starting to write.

Calculating the number of characters in the receive buffer (`no_rx_chars`) is very similar to the previous one. This is very useful where the communications are packet-based as the applications software can wait for a particular number of characters to be received before starting any processing.

Finally

Figure 2 is a simple terminal emulator written in assembler. Even if you don't understand UARTs, the code presented in Figure 1 should make a useful library module which can be called from high-level languages that can use C calling conventions.

EXE

John Davies has worked for several years in a variety of real time embedded software applications. The code accompanying this article is available on disk. Send a blank formatted floppy disk to the Editor, following the instructions given on Page 1, column 1. Mark your envelopes 'PCSERIAL'.

```
  NAME TERMINAL
  ASSUME CS:_TEXT,DS:_TEXT,SS:STACK
  _TEXT SEGMENT PARA 'CODE'
;****************************************
; Extenal procedures
; -----------------
  EXTRN   _start_serial_ints      :FAR
  EXTRN   _stop_serial_ints       :FAR
  EXTRN   _send_sio               :FAR
  EXTRN   _read_sio               :FAR
;****************************************
; Terminal software
; ----------------
  MOV AX,CS
  MOV DS,AX
; Set up serial interrupts
  MOV AX,0E3H              ; 9600 baud,
                           ; 8 data,1
stop
  PUSH AX
  CALL _start_serial_ints
  POP CX                   ; Clean stack
start:
; Read character from keyboard
  MOV AH,6
  MOV DL,0FFH
  INT 21H
; Check if a key has been pressed
  JZ check_read
; Check for ESC pressed
  CMP AL,1BH
  JE finish                ; Yes=termi-
nate
; Key pressed, character in AL,
; write to output
  MOV AH,0
  PUSH AX
  CALL _send_sio
  POP CX                   ; Clean stack
; Check for chars received
; - until ESC is pressed
check_read:
  CALL _read_sio
  CMP AX,0FFH
  JG loop_back             ; No chars
; Display received character
  MOV AH,2
  MOV DL,AL
  INT 21H
; Loop back
loop_back:
  JMP start
; Terminate program
finish:
  CALL _stop_serial_ints
  MOV AX,4C00H             ; And exit
  INT 21H
  _TEXT ENDS
;****************************************
  STACK SEGMENT STACK 'STACK'
  DB 256 DUP (?)
  STACK ENDS
;****************************************
END
```

Figure 2 - Terminal emulator in assembler

Deskterm™

Discover how you can add a Motif Interface onto Character-based Applications.

YOUR RECIPE FOR SUCCESS

COOKING TIME

90% saving on re-writing your software for X

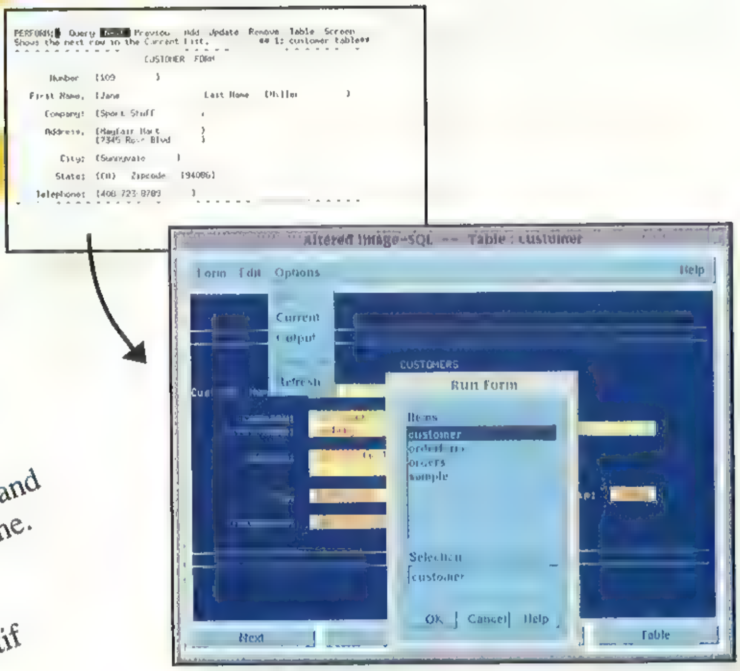
METHOD

Take existing character-based application

Add IXI's Deskterm Software.

Run on UNIX system and simmer for a short time.

Before serving, decorate with Motif



Like many software developers, you may be evaluating how to produce a Motif version of your application. It's no easy decision. For many businesses, the task of rewriting a 20 year old COBOL application or one written in FORTRAN or a 4GL is too costly in terms of time and resources.

IXI has the answer. Deskterm will revolutionize your path to Motif. Experience shows that Deskterm can reduce the time to add a Motif front-end by more than 90%, bringing your application to market years ahead of the competition.

Using Deskterm you can produce a fully featured graphical user interface with pull down menus, scroll bars, dialog boxes, pushbuttons, mouse control and multiple fonts and colors. What's more, you can do all this without access to the application's source code.

You don't need to know anything about X programming (at either Xlib or toolkit level) to move your software to Motif. Deskterm uses existing character-based programming skills allowing you to concentrate your time and money on building applications rather than wrestling with X and toolkits.

UNIXWORLD 1·9·9·0 best PRODUCTS

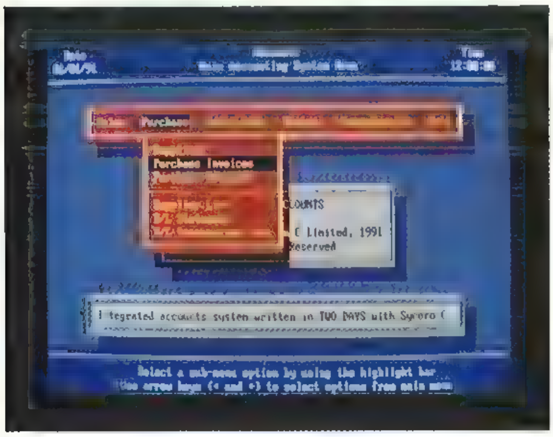
So if you're looking for a fast and painless route to Motif, contact us today for information on +44 223 462131.

IXI

IXI Limited · 62-74 Burleigh Street · Cambridge · CB1 1QJ · England · Tel: +44 223 462131 · Fax: +44 223 46132

CIRCLE NO. 584

SoftwareDevelopers!



Produce a sophisticated application like this in Microsoft C , with either BTRIEVE or C-ISAM, running under DOS in just a couple of days!

...Here's the same application half an hour later usingTurbo C ++'s advanced VROOM technology...

...Another twenty minutes and it can access dBASE files...

...and here's the same application an hour later running under SCO UNIX, with no runtime licences required !

This entire application was developed in a couple of days using **Sycero C**. Sycero is a powerful program generator that **dramatically** reduces your programming time. Sycero C generates fully structured and documented C source code to be compiled with Microsoft or Turbo C. Sycero C supports the Btrieve/C-ISAM file handling systems, plus it can even generate C programs for use with dBASE or Clipper files and indexes. Sycero C UNIX allows you to develop software to run under SCO UNIX, and you can regenerate existing Sycero C DOS-based applications with absolutely **no changes** required to your definitions.

For more information, either call us on **0622 691616**, or fill out and return the coupon.

CIRCLE NO. 585

dBASE is a trademark of Ashton-Tate. All other trademarks acknowledged.

Please send me further details about Sycero C

Name: ....

Company: ....

Address: ....

....

....

....

Tel: .... Fax: .... EXE 3/92

I am also interested in your generator for Clipper, Sycero dB ☐

System C Ltd 60-61 High Street Maidstone Kent ME14 1SR

Tel 0622 691616 Fax 0622 691241

More Paradoxes

Larry Adlard blows the lid on Paradox's proprietary file system.

One of the components of many systems we write is a high speed real time order entry program. When a customer phones in, the person he talks to has on screen all the details of his account, access to all stock records, and the customer's past history. The intention is to take the order there and then, with delivery notes, invoices and any other documentation being produced within a few seconds. All records are updated and the job is completed in one pass.

Paradox is a fine product but, unfortunately, the world is not perfect and in spite of my full and enthusiastic recommendation, Paradox has its limitations. When it comes to the kind of speed we sometimes need, Paradox just simply can't cut it. That's not a criticism, just a recognition of reality. What that means is that the front end of the system is going to be written in another language and in our case it will be assembler.

Mixing two 'languages' can cause complications but in our case the assembler program is one standalone thing. The data it operates on is maintained in and by the Paradox system. This means the format of the data can be rearranged by the customer, data can be added and deleted and the assembler program adapts to the revised data format.

The crux is that in order to work this way you have to know how Paradox organises its files and as far as I know this information is not published anywhere. Borland (née Ansa) keep the method confidential. If you already write in C, then Borland will now sell you a library of routines for access, but this was not available in 1987 when we needed it and we don't particularly wish to write in C. If you can accommodate the Paradox Engine then this may be your best solution and you will note that I am bending over backwards to point out the alternatives before encouraging you to try and manipulate Paradox files directly. We don't go out of our way to break the rules, but needs must when the devil drives.

If despite the alternatives, you find yourself in a similar position to ourselves, then the following information will allow you to read and write data reliably, to and from Paradox files, subject to the following house rules.

DO NOT attempt to create or extend Paradox files. Let Paradox create its own file and pre-format as many records as you are likely to need. If necessary store records in a temporary Paradox file and let Paradox append these to the main database.

DO NOT attempt to maintain indexes or password encrypted files unless you are a masochist.

Paradox files are similar in principle to dBASE files, ie the data file is self documenting. The data is preceded by a header which encapsulates all the information required to decode it. There the similarity ends. Using DOS debug, anyone can inspect a Paradox file but remember debug loads the file at 100h for execution as a program, so all the address references are inflated by 100h. Figure 1 shows the file as it is. The Paradox structure is shown in Figure 2.

The first word is in standard Intel, low-byte first format and represents the length of the record (0066h) = 102 decimal bytes in this case.

The second word is the address where the data actually starts (0800h).

The third word is marked in our files as the unknown soldier; every file we've seen has the value (0202h). Since we don't remember and no longer have access to any Paradox version 1 files, we suspect but cannot confirm that this represents a version number.

The fourth word (01ECh) is the current number of records stored (492).

The fifth word has no use known to us.

The sixth word gives the file length in Paradox blocks or packets (26).

Moving on, the 34th byte (021h) gives the number of fields. One byte is large enough because 255 is the maximum.

If at this point you are operating legally, have access to Paradox, and have created the file you have almost everything you need. The only other aspect you need to know about is the technique which Paradox uses to place records into packets which is described later. In case you need a little more detailed information, a brief discussion of the descriptors follows.

Curiouser and Curiouser

At location (0030h) there are two four-byte patterns. These are not relevant until after you understand the field descriptors but the patterns they form should be noted. Field descriptors start at (0058h). These are two-byte pairs. The first byte indicates what type of field it is (See Figure 3), the second byte gives the field length in bytes (again maximum 255). With the exception of Alpha-Numeric Fields all the other types have a predefined length. A record can have between 1 and 255 fields, therefore the area which contains the field descriptors can vary in length from 2 to 510 bytes. In the example shown there are 18 fields, so the field descriptors are 36 bytes long. Following immediately after this is a four-byte pattern followed by another 18 four byte patterns. These are similar to the other two noted earlier.

The first four-byte pattern after the field descriptors is a pointer to the Filename at location (00C8h). The next one points to field heading 'Carma' at the location (011Fh). This is the name that Paradox knows that field by. The remaining four-byte blocks each point in turn to the next fieldname until all fields have been named. Immediately following that is the Filename.

By now the four-byte pointers may be puzzling you since they do not point directly to the data addresses in your file. The values are so large that their point of origin is way outside the data file. Bearing in mind that Paradox files are intended to be accessed from within the Paradox program (and without any authority whatsoever) we might speculate that these are an offset from a fixed data location within the Paradox program itself. All is not lost however, the difference is a constant. You will note that the last two bytes are common throughout and can therefore be ignored altogether. The first two bytes when added together also do not directly point to the data they represent but the error is now much smaller. For example the pointer to the Filename (007Ch) reads 3E 8E 77 1B. Adding the first two bytes together, the result is (3E+8E=CC) but the Filename isn't located at CCh, it is located at C8h, a difference of four. Now that you know the first two bytes are wrong by a displacement of four, lets try the next block. This reads 8D 8E 77 1B. Adding 8D to 8E gives 11Bh less the four bytes constant error gives 117h. Lo and behold we have calculated the offset from the beginning of the file to the name of the first field. Once the difference has been established it can be applied to each pointer in turn until all the field names have been recovered.

```
Addr:           H E X               ASCII                    DECIMAL
0000: 66 00 00 08 02 02 EC 01 h....... [ 0102 ]  [ 2048 ]  [ 0514 ]  [ 0492 ]
0008: 00 00 1A 00 1A 00 01 00 ........ [ 0000 ]  [ 0026 ]  [ 0026 ]  [ 0001 ]
0010: 1A 00 23 00 00 00 00 00 ..#..... [ 0026 ]  [ 0035 ]  [ 0000 ]  [ 0000 ]
0018: 00 00 32 03 6F 1B 00 00 ..2.o...  0     0   050  003  111  027   0    0
0020: 00 12 00 00 00 00 00 00 ........  0   [018]  0    0    0    0    0    0
0028: 00 B7 00 00 00 7B 7A 00 .....{z.  0    183   0    0    0   123  122   0
0030: F2 8D 77 1B CE 8D 77 1B ..w...w. 242  141  119  027  206  141  119  027
0038: 00 04 1A 00 00 00 1F 0F ........  0   004  026   0    0    0   031  015
0040: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
0048: 00 01 00 00 00 00 00 00 ........  0   001   0    0    0    0    0    0
0050: 00 76 01 00 00 00 00 00 .v......  0   118   0    0    0    0    0    0
0058: 01 08 01 01 01 20 01 01 ........ 001  008  001  001  001  032  001  001
0060: 05 08 05 08 05 08 05 08 ........ 005  008  005  008  005  008  005  008
0068: 03 02 03 02 03 02 03 02 ........ 003  002  003  002  003  002  003  002
0070: 06 08 06 08 01 01 01 01 ........ 006  008  006  008  001  001  001  001
0078: 03 02 01 02 3E 8E 77 1B ....>.w. 003  002  003  002  062  142  119  027
0080: 8D 8E 77 1B 93 8E 77 1B ..w...w. 141  142  119  027  147  142  119  027
0088: 95 8E 77 1B 9B 8E 77 1B ..w...w. 149  142  119  027  155  142  119  027
0090: A0 8E 77 1B A6 8E 77 1B ..w...w. 160  142  119  027  166  142  119  027
0098: AC 8F 77 1B B2 8E 77 1B ..w...w. 172  142  119  027  178  142  119  027
00A0: B8 8E 77 1B BE 8E 77 1B ..w...w. 184  142  119  027  190  142  119  027
00A8: C4 8E 77 1B CA 8E 77 1B ..w...w. 196  142  119  027  202  142  119  027
00B0: D0 8E 77 1B D6 8E 77 1B ..w...w. 208  142  119  027  214  142  119  027
00B8: DC 8E 77 1B DE 8E 77 1B ..w...w. 220  142  119  027  222  142  119  027
00C0: E0 8E 77 1B E6 8E 77 1B ..w...w. 224  142  119  027  230  142  119  027
00C8: 52 45 53 54 54 45 4D 50 RESTTEMP  R    E    S    T    T    E    M    P
00D0: 2E 44 42 00 00 00 00 00 .DB.....  .    D    B    0    0    0    0    0
00D8: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
00E0: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
00E8: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
00F0: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
00F8: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
0100: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
0108: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
0110: 00 00 00 00 00 00 00 43 .......C  0    0    0    0    0    0    0    C
0118: 61 72 6D 61 00 54 00 44 arma.T.D  a    r    m    a    .    T    .    D
0120: 65 73 63 72 00 50 61 63 escr.Pac
0128: 6B 00 42 61 6E 64 31 00 k.Band1.
0130: 42 61 6E 64 32 00 42 61 Band2.Ba
0138: 6E 64 33 00 42 61 6E 64 nd3.Band
0140: 34 00 41 76 61 69 6C 00 4.Avail.             et cetera
0148: 52 65 73 76 64 00 4F 6E Resvd.On
0150: 4F 72 64 00 53 74 4D 69 Ord.StMi
0158: 6E 00 42 61 53 65 50 00 n.BaSeP.
0160: 43 6F 73 74 50 00 58 00 CostP.X.
0168: 59 00 54 72 43 6E 74 00 Y.TrCnt.
0170: 53 70 61 72 65 00 00 00 Spare...  S    p    a    r    e    0    0    0
0178: 00 00 00 00 00 00 00 00 ........  0    0    0    0    0    0    0    0
 --     -----------
0800: 02 00 00 00 92 07 31 30 ......10 < 6 byte header followed by 1st Record
0808: 30 31 46 31 42 4B 54 53 01F1BKTS   A8 + A1 + A32
0810: 50 4F 4F 4C 45 44 20 31 POOLED 1
0818: 30 4D 20 42 4C 41 43 4B 0M BLACK
0820: 00 00 00 00 00 00 00 00 ........ < Note padding of Alpha string with 0
0828: 00 00 00 00 00 00 00 4C .......L < Line ends with A1 Alpha field
0830: BF E4 7A E1 47 AE 14 7B ..=p...= < All 8 bytes are a currency value
0838: BF E7 0A 3D 70 A3 D7 0A ...=p...            ditto
0840: BF E4 7A E1 47 AE 14 7B ..z.G..{            ditto
0848: BF E3 33 33 33 33 33 33 ..333333            ditto
0850: 80 08 80 00 80 00 80 90 ...  ... < 4 x 2 byte short integers
0858: BF DA 7E F9 DB 22 D0 E5 ..~..".. < All 8 bytes are a numeric value
0860: 00 00 00 00 00 00 00 00 ........            ditto
0868: 00 2A 00 00 31 30 30 31 .*..1001 < 2 x flags + A2 Spare + Next Record
0870: 46 31 42 52 54 53 50 4F F1BRTSPO
0878: 4F 4C 45 44 20 31 30 4D OLED 10M
```

Figure 1 - A memory dump of a Paradox file header

Even if the Filename wasn't visible it is possible to obtain its address. The file descriptors start at the fixed offset 0058h. The byte at 0021h gives the number of fields and each field has a descriptor that is two bytes long. Beyond that area there is a block of four-byte pointers to each field title plus a four-byte pointer to the Filename. Given N fields, the address for the start of the Filename can be calculated from: 0058h+((2*N)+((N+1)*4)). This value can be compared to the Filename pointer to obtain a correction factor which then allows you to find the field names, which because of the variable length of the field descriptors, could be anywhere.

Unless you are going to write a better utility for reading Paradox files than the people who wrote Norton Commander, you won't need most of this information. If it is your file you will already know what the fields are, and what type and size they are. You will however need to know how Paradox places records into packets, as described later.

Paradoxical Packets

It always used to be our practice to try and make record lengths binary multiples such as 16/32/64/128 bytes. The reason is purely practical and dates from when computers were much less powerful. Hard disks don't read records, they read sectors of 256 or 512 bytes. DOS reads multiple sectors into blocks. If you have an 'odd' length record of say, 117 bytes, then depending on sector size and block size a record will eventually straddle a block or sector. The performance penalty is that two blocks or sectors have to be read to obtain first one portion of the record and then the remainder from a second block. Depending on the way the machinery has been configured it could take 3 or 4 times as long to retrieve that record.

The designers of Paradox obviously despairing of efforts to persuade application programmers to write 'even' length records arranged to store records in packets. This practice can work against efficient storage unless you are aware of it. A data packet can be 1 KB, 2 KB or 4 KB long and which size it is depends on the record length.

The first six bytes of each packet are used for important housekeeping functions which are explained later. If say, your record is 128 bytes long and Paradox chooses

How much longer can you afford to wait?

Create Overlaid Programs-Fast.

BLINKER™, the world's first and fastest dynamic overlay linker, reduces your link time to seconds and reduces program memory requirements. Now you can use one linker for all your software projects.

One Linker, Many Languages.

BLINKER 2.0 links and automatically overlays DOS programs written in Microsoft® C, BASIC, Assembler, QuickBASIC™, Fortran, Pascal, Watcom™ C, Zortech C++, Clipper®, FORCE® and in Borland® C, C++, Assembler, and more.

Save Time and Memory.

BLINKER removes the need for overlay structures, simplifies program design and reduces memory requirements to save you time, effort and memory.

Memory Swap Function.

BLINKER is the ONLY linker to offer an integrated memory swap function, so you can run other large programs from within your program, with negligible memory overhead.

BLINKER™
High Performance Dynamic Overlay Linker

Don't Settle for Less.

Other major features include full CodeView® support, use of EMS/XMS at program run-time, and enhanced execution speed of overlaid code.

Time is Money.

BLINKER offers all this in a fraction of the time it takes to link with your current overlay linker. You know time is money, and link time is no exception.

Free Demo

To try our free demo on your own code and for more information contact our U.K. distributor:

Call: +44-81-994-4842 or
FAX: +44-81-994-3441

QBS Software Ltd.
10, Barley Mow Passage
London W4 4PH

BLINKER is available in 5.25" or 3.5" diskette format

Price £199
plus shipping & handling

VISA. MasterCard

Blink*inc*
P.O. Box 7154
Richmond VA
23221

CIRCLE NO. 586

STRUCT	Field Name	Field Type	
1	Carma	A8	This is the product code
2	T	A1	Flag for internal manufacture
3	Descr	A32	Full description
4	Pack	A1	Pack type
5	Band1	$	}
6	Band2	$	} Banded
7	Band3	$	} Prices
8	Band4	$	}
9	Avail	S	Qty available for sale
10	Resvd	S	Qty reserved
11	OnOrd	S	Qty on order
12	StMin	S	Stock Minimum
13	BaSeP	N	Base Price (minimum selling price)
14	CostP	N	Cost Price
15	X	A1	Flag to indicate a price change
16	Y	A1	Flag to indicate Stock Minimum warning
17	TrCnt	S	Transaction count
18	Spare	A2	Example of our internal padding

Figure 2 - The Paradox Structure Table

a packet size of 2048 bytes then you would expect each packet to contain exactly 16 records but Paradox needs 6 bytes of each packet and it NEVER, NEVER allows a record to straddle two packets. What you will get for your 2 KB is a six byte header 15 complete records and a wasted area of padding 122 bytes long. If on the other hand your record is 127 bytes long you will get the 6 byte header 16 records and only 10 bytes of padding. The worst cases occur with really long records. (See Andy Redfern's article *Performance Tuning Paradox* - *.EXE* April '91.)

Paradox effectively trades disk space for speed if you choose an inappropriate record length. Paradox's true love is exactly 15 bytes long, unfortunately not many customer names and addresses fit easily into this length, so sometimes a compromise has to be struck.

You will note that the record length in this case is 102 bytes long. Using our previous system of binary multiples this would be terrible. Paradox will place records of this length into 2 KB packets so in a 2048 byte packet it will fit 20*102 byte records (2040) plus a six byte header and only two bytes will be wasted. Every DOS read retrieves 20 records.

Christie Thriller

If you have followed every detail so far you may be grateful to reach the end. Like an Agatha Christie novel, there is one more little twist. The six byte header at the beginning of each packet is actually three words. The first word is a pointer to the packet (the packet number) which contains the next record. The next word is a pointer to the packet which contains the record previous to the first in the current packet. The third word is a pointer to the beginning of the last valid record in the packet. (Packet zero is the header.)

Observing the specimen file, the first word (0800h) has the value two. Since all the records were entered sequentially, this is as one might expect. The first packet contains 20 records and the 21st record is the first record in packet number two. If we delete record number two all the records after number two in the first packet are promoted 3>2 4>3 etc. The third word in the header is reduced by one record length (0792h-66h=072Ch) and the number of records at (0006h) is reduced by one. A copy of the record which was at position 20 in the packet and was copied forward to position 19 remains in position 20 but never appears in Paradox because the byte count in the third word of the packet header stops Paradox from reading beyond the 19th record.

Assuming the file is returned to its previous state consider what happens when a record is inserted. We insert a new record at two. All the records will be demoted this time by one position but the packet can only contain 20 records so the record previously at position 20 will now 'fall off' the end of packet one. There are two possible ways of dealing with this. The first solution is to continue demoting every record in the file until the whole file has been extended. You will be sorry to learn that Paradox doesn't do that. Paradox creates an entirely new packet and increments the packet count at location (000Ah). Since there are 26 packets already, the new packet is appended to the end of the file. The three words at the beginning of the new packet will read 0001h 0002h 0000h. The first word indicates that the previous record is the last valid record in packet number 1. The second word tells you that the next record in the sequence (after all the records in this packet) is the first record in packet number two. The third word indicates that only one record is present in this packet since the last record in the packet starts at offset 0000 (+6 bytes for header). In effect, when a packet is full and a new record is inserted, the overflow is placed into a new packet which forms a semi-'B Tree' like structure. From now on this new packet is reserved for overflow from packet number one. Paradox does not fully initialise the new packet. If you look at it you will see the six byte header of the record and the remainder of the 2 KB may be garbage. Alternatively, it may contain the .BAK version of the managing director's reply to Personnel, regarding your request for a raise.

If you now delete the recently inserted record number two, all the records in packet number one will be promoted again but the overflow record in packet 27 will not be moved back to block 1, it will remain where it is. As you add records the file gets longer. If you delete records the space is left vacant. Empty nodes are not permitted, so the file never gets shorter. I know of only one way you can recover space from, or defragment a Paradox file. You have to create a new file with an identical structure to the old one and read the records one by one from old to new. You can then delete the old and rename the new to the old name.

It may not have escaped your attention that the file is empty between locations 0200h and 07FFh. Partly this is to make the header up to packet size but do not be mislead into thinking it might be available for your use. In certain circumstances Paradox does use this area and if it does, the methods outlined here are likely to be entirely inappropriate. The value at (0030h) does in fact point to a location in this area which ensures that the field titles will not be overwritten.

Without giving any secrets away, you might be aware that Paradox files can be password encrypted. The techniques outlined above will not enable you to circumvent this protection, nor would we publish a

No	Type	Length
1	Alpha-numeric	1 to 255
2	Date	4
3	Short	2
4	Long Integer	4
5	Currency	8
6	Numeric	8

Figure 3 - Paradox field types

5 out of 5 hackers prefer other software protection methods to Hardlock E-Y-E®

What hackers dislike...
Hardlock E-Y-E was designed using cryptographic principles. It took the experience and know-how of Germany's No. 1 in software protection and the leading edge technology of a US semiconductor company to create the ultimate software protection tool. Hardlock E-Y-E is based on a custom chip featuring secure algorithmic response rather than simple bit swapping or counting schemes.

What software developers like...
Hardlock E-Y-E combines all the features software developers require in a single product: algorithmic response to provide security and an optional non-volatile memory to allow custom configurations. FAST Electronic has made implementation of Hardlock E-Y-E in your software easy. Use HL-Crypt to protect .EXE or .COM files, or incorporate high level language interface routines in your software. The algorithm parameters and the contents of the memory can be programmed in seconds using our Crypto-Programmer card. This unique card guarantees that no one else can burn your original codes. Simply plug the card into any PC slot and start up your own Hardlock E-Y-E workshop.

What your customers will like...
Hardlock E-Y-E allows unlimited backup copies. The device is shipped with the software for the user simply to plug into the parallel interface and forget. Daisy chainability, outstanding reliability (no battery is needed), and the most compact High-Tech design ensure that your customer will accept Hardlock E-Y-E.

What your accountant will like...
Hardlock E-Y-E needs no factory coding. This ensures optimum delivery schedules and stock flexibility. Revenues will go up as software piracy and multiple usage are prevented. Despite its wealth of features, Hardlock E-Y-E's prices remain competitive.

Hardlock E-Y-E
programmable, algorithmic response and memory option – all in one.

...As more and more software developers, customers and accountants appreciate the Hardlock E-Y-E device, hackers like it less and less.

MAGNIFEYE

Order your demo unit today. Contact Magnifeye,
235-239 Walmer Road, Walmer Studio #6, W11 4EY, Telephone 071 221 8024, Fax 071 792 3449.

CIRCLE NO. 589

method for doing so. We never use the feature ourselves because it only addresses half of a problem. If you password encrypt your files then you had better be certain you have 100% accurate and reliable backups. If you can't guarantee a reliable backup copy of your data and your hard disk suffers from a wordslip, then not only will access to your data be denied to others, you won't be able to access it either! That is not to say the encryption system is unbreakable, especially if you know what the data is supposed to look like, but the encryption method is sophisticated and cracking it is not a trivial matter. The information concerning the structure of the file is not affected by encryption although the data is. If you genuinely believe your information is sensitive enough to justify encryption then it also justifies the cost of a reliable, dedicated backup system to match.

Skinning Cats

Another feature of Paradox files is that you can create secondary key indexes using the {Tools}{Query}{Speedup} function. This creates a pair of index files with extensions (.X01 and .Y01). These indices are also kept in the same file format as the Paradox data files but re-sorted. The techniques outlined do not attempt to maintain these indexes, although you will be able to access the data file on a read-only basis. Again our experience indicates that there are alternative ways to obtain performance without recourse to indexing files.

Field 17 `TrCnt` is a transaction count. Every time an item is sold this value is incremented. Every time Paradox is loaded, it looks for a script called 'Init' and executes it. It can be made to accept a password or to sort a file. If in our case, it sorts the file so that items with the highest transaction count are at the beginning of the file, then the items which are needed most frequently will have the shortest access times, furthermore, as demand patterns change the system adjusts. By this means we save the storage space of the two indexes and have the most efficient algorithm available for our purpose. Far from adding to the system requirements this feature gives vital management information. The ten items at the top of the list are the 10 best sellers. The ten at the bottom are candidates to be dropped from the product range. Even that isn't the end of it. Many manufacturers will produce items to order which they don't normally have a demand for. In this example, stock minima are re-calculated as a moving average of demand over the past six months. It is important that the stock minimum is not increased on the basis of a single large order. So our stock minimum calculation program will be 'moderated' by the number transactions. A single order for say, 2000 items will not alter the stock minimum from 0 whereas 100 orders for 20 will.

Conclusion

Paradox files can be directly manipulated - if your need is great enough.

EXE

Larry Adlard is the managing director of A & A Management Research Ltd. He originally trained as an accountant but switched horses when he decided that programmers, not accountants would rule the earth. He has been involved in the 'Micro business' since 1978. The first machine he programmed had 2KB of memory, a hexadecimal keyboard, a three and a half digit, seven segment LED display, and an audio tape drive interface! History records that it actually worked.

Memory

Some UNIX systems can now take advantage of virtual paging to map files into memory. Peter Collinson has been using the technique to soup up some tired old code.

I have been doing some 'real' programming recently, porting some code that I wrote around five years ago. The code is a cooperating suite of 6 programs, around 25,000 lines of C. Its job is to generate the quarterly invoices and statements for the UK part of Usenet, UKnet. It turned out that it was sensible to rewrite most of the code to adapt to the world now, rather than the world that existed when the programs were originally created.

The exercise has lead me to think a little about how I go about doing things and how that has changed since I started writing programs for UNIX in 1976 or so.

Some background

The accounting programs are not particularly interesting in themselves. I guess that they are 'very UNIX' in the sense that they are designed to read information from many different sources, and these sources are all text files.

A major part of the input data is a set of files called 'the maps' which are used to store information about each individual site on the network.

Each file contains lines of the form: a keyword, a colon, and some data. The map entry for my site begins:

```
Name: hillside
NRS-Name: uk.co.hillside
Organization: Hillside Systems
Contact: Peter Collinson
Phone: +44 227 761824
Fax: +44 227 762554
Postal: 61 Hillside Avenue,
        Canterbury,
        Kent CT2 8HA
Electronic: pc@hillside.co.uk
...
```

By convention, my map entry is stored in a file called `hillside`. This makes it easy for the people maintaining the information to find the relevant file. The programs make no use of the filename, they know that every file contains a record for one site.

Processing the maps

The accounting programs need to read data from these maps. Different programs will need to read different subsets of the data. For example, consider the program that looks in the mail logs to work out who has sent mail and should be charged for the privilege. It needs to know the name of each site that it will charge. Sites also have aliases to their names, the `NRS-Name` is one of these. The program must pick this aliasing knowledge from the maps. It needs this information randomly, since it is dealing with inbound mail which is completely unordered. On the other hand, it doesn't need to know the name and address of the person who will eventually receive the invoice, only the program that generates the PostScript invoices and statements needs to know that.

If I was implementing this for an early UNIX system, I would have worried about the memory that was available to my program. Life was simpler then and we would have been dealing with many less sites than the present 750 but let's forget that for a minute. If you concatenate all the data for the current set of sites you get a file that is around 87 KB. There's no way that we can read all that information into the limited memory space of our early UNIX system (limited to 32 KB) so we have to adopt different strategies.

The programs would only read the information that was needed for them to operate. The mail charging program would perhaps be able to scan the maps and build up a table of names in memory. The invoice printing program needs a lot of information from each map, but will need it in less random order. It can afford to read the map that it requires for the site that it is processing at the moment.

Both of these programs will read the map files perhaps using the standard I/O library. They will read data from the files a small piece at a time, storing what is needed for later processing and simply discarding other stuff. The interesting thing about this approach is that it has become a standard, a little *de facto* in places, but a standard never-the-less. It is portable into a great number of different environments, from the weakest PC based system, through UNIX workstations, into small mainframe systems running a variety of operating systems and up to to the largest IBM based installations.

Pulling files into memory

When I moved from the memory limited systems onto the systems that supported virtual memory, I began to stop using the 'read pieces into memory' approach. The original accounting package of five years ago was coded using a routine that pulls a complete file into memory:

```
typedef struct {
  char *base;
  int  len;
} Fhandle;

Fhandle *
load_mem(fname, fp)
  char *fname;
  Fhandle *fp;
{
  int fd;
  int sz;
  char *rv;
  struct stat statb;

   fd = open(fname, 0);
   if (fd < 0)
    return(NULL);

  fstat(fd, &statb);
  sz = statb.st_size;

  rv = malloc(sz);
  if (rv == NULL)
    sz = 0;
  else
  if (read(fd, rv, sz) != sz){
    free(rv);
    sz = 0;
  }

  close(fd);
  fp->base = rv;
  fp->len = sz;

  return(sz != 0 ? fp : NULL);
}
```

Special Low Price offer to *.EXE* readers

The Copam®

386DX-33 Computer

(Upgradable to 486DX)

Low Radiation monitor

... and DOS 5.0

... and Windows 3.0

... and Mouse

Features

- Intel 80386DX 33MHz. Provision for a 80387DX co-processor
- 64Kb fast 25nSec SRAM cache
- 2-16 Mb RAM, 0 wait state
- Shadow RAM for the BIOS
- 1.44 Mb and/or 1.2 Mb Disk Drives
- 40 Mb, 100 Mb, 180 Mb Hard Disk
- Two RS232C Serial Ports
- One parallel Centronics printer port
- 102 Key U.K. keyboard
- 16 Bit Super VGA card 1024 x 768, 0.5Mb or 1Mb memory
- Low Radiation, Super VGA colour monitor
- Microsoft DOS 5.0
- Microsoft Windows 3.0
- Three-buttons Serial Mouse
- Copam's one year r.t.b. warranty (on-site maintenance optional)
- Barnett Electronics' *(est. 1979)* second-to-none after sales service
- 10 Days trial. Money back guaranteed if returned within 10 days

intel inside

For price and further information please contact:

BARNETT ELECTRONICS LTD
10 Barley Mow Passage, London W4 4PH

081 995 3715

CIRCLE NO. 591

Of course, this omits loads of include files and doesn't worry about casts and other bits of current C bag and baggage. It should test for zero length files and worry about the return values of various calls.

The routine opens the file and uses `fstat` to find its size. The `malloc` routine grabs some memory and a single `read` call is used to pull the file into memory. I have wrapped this up using a plausible call mechanism. I guess that I generally will sprinkle the routine itself with failure messages where needed. The actual details of this interface change every time I write this code.

The idea here is that you can get the whole file into memory by calling

```
rv = load_mem("file", &fh);
```

This revolutionises programming. Suddenly, many of the problems associated with reading the file line by line or character by character simply disappear. It always seems much less complicated to look at a bunch of characters in memory and parse them than it is to read them in one at time doing the same job. I generally write code that zips through the connected file looking for strings and creating a table of structures. Perhaps:

```
struct address {
  char *contact;
  char *organization;
  char *postal;
};
```

These variables will point directly at the memory containing the file. I tend to ensure that the values become zero-byte terminated strings because it makes life easier. This is easy, simply replace the newline character that ended the line in the file by a null byte and off we go. We have to worry a bit about extraneous white space.

The routines that extract the information from the file that has been read into memory are often pretty disgusting with much ad-hoc pointer handling. But they do work quickly, and that is another desired aim.

Again, this technique is reasonably portable if you have enough memory. If you run out of memory to allocate for the file, then you have probably blown it - the program will just fail.

Mapping into memory

The technique of allocating space and pulling files into memory using one read call is reasonably fast. It does mean that the data is copied. At best, it comes from the disk into kernel memory and is moved from there into user space. On machines with demand paging, it should be possible to use the paging system to bring data files into memory. After all, that is what is happening to the executable bits of programs that are running. This idea was a gleam in the eye of the designers of 4.2BSD in the mid-80's. They specified the system call and how it should operate. Unfortunately, it was never implemented because of deficiencies in their paging system.

In recent times, Sun Microsystems implemented the call and this has found its way into System V, release 4. It's also being defined by the working group looking at real time extensions in POSIX. I expect that the ability to map files into memory will be present in all UNIX systems within a year or so.

Let's rewrite the `load_mem` routine in terms of the memory mapping system call `mmap`.

```
Fhandle *
load_mem(fname, fp)
  char *fname;
  Fhandle *fp;
{
  int fd;
  int sz;
  char *rv;
  struct stat statb;

  fd = open(fname, 0);
  if (fd < 0)
    return(NULL);

  fstat(fd, &statb);
  sz = statb.st_size;

  rv = mmap(0, sz,
         PROT_READ|PROT_WRITE,
         MAP_PRIVATE, fd, 0);

  if ((int)rv == -1)
    sz = 0;

  close(fd);

  fp->base = rv;
  fp->len = sz;

  return(sz != 0 ? fp : NULL);
}
```

Again the code needs some improvements for real use. The first part of the routine is the same, the file is opened and its size obtained. The `mmap` routine is then called. This call returns the base address that has been allocated for the file or -1 on any error.

The mmap system call

There are six parameters to the system call. Beware that the information here pertains to Sun's implementation and things might be different on your system.

```
caddr_t mmap(addr, len, prot,
                  flags, fd, off)
caddr_t addr;
size_t len;
int prot, flags, fd;
off_t off;
```

The first, `addr`, tells the system where the process would like to have the file mapped into its address space. Most of the time this is 'don't care' and a zero value says just that. If you wish, you can supply a specific address that acts as a positioning 'hint' to the kernel. The file will be connected at a convenient address near to the hint. If you add `MAP_FIXED` to `flags` then you can force the value of `addr` to be the base address for mapping. This is not recommended since you might adversely affect the way of the system wishes to manage its resources.

The second parameter, `len`, gives the length in bytes that are to be mapped by the call. It doesn't have to be the length of the file. You should think of the map as a 'view' into the file; and views can be altered. If `len` is longer than the file, then the remaining bytes up to the next page boundary are accessible and are zero filled. Your process will be sent a SIGBUS signal if you attempt to access above that.

The third parameter `prot` gives the form of protection that will be given to the pages allocated for the file. You can supply read, write and execute protection. In the case above, you might think that we only want to read the file and should say `PROT_READ`. You will recall that we want to stuff null bytes into the data, so we want to write to the pages. We will also use `PROT_WRITE` to say that we are going to write to the memory that maps into the file.

The `flags` parameter provides other information about handling the mapped pages. A subfield of the `flags` parameter indicates options that are applied when pages are altered by the program. Sun gives you two options. `MAP_PRIVATE` indicates that a change to the page is to be retained by the process. By setting this in the example, we ensure that our null bytes don't find their way out onto the file, the pages stay private. `MAP_SHARED` says that a change to the page will be reflected in the actual stored file and also in the data that any other process reads for the file. `MAP_FIXED` can be ORed into the parameter.

The `fd` parameter is the file descriptor of the open file. The `off` parameter is a file offset, this allows you to change the view on the file, mapping the file into memory starting from this offset.

Mapping actions

Nothing visible happens when the file is mapped into memory. Actually the page tables for the process will be altered to point at the particular file, but the file will not be read. The page table entries will say that the data is paged out to disk. As soon as the process accesses the data in the mem-

ory, the hardware will force a page fault and the full panoply of the memory management system will be brought to bear on the problem. The disk will be accessed, the page read into memory, the page tables fixed up and the process restarted.

None of this is apparent to the process, it sleeps until its data is available. If the process doesn't touch a page in the file then the data is not read. This can be a win if you only need to look at parts of the file.

Moving data into the pages constitutes a write operation to the file. We have seen that you must enable writing by supplying `PROT_WRITE` in the `prot` parameter. If the `flags` parameter contains `MAP_PRIVATE` then a change that has been made to the page is private to the current process. To achieve this, the original page is copied in memory and the page table entry for the process is set to point at the new page.

If the `flags` parameter contains `MAP_SHARED` then the change that is made to the page of the file will find its way onto disk. The contents of the page may be pointed to by several processes and all these processes will see the new page contents.

Unmapping a file is simple, you just call `munmap()`. The call releases any memory resources that are used by the mapping. To unmap the file, all we need is the base address and a length. The system call to do this looks like:

```
int munmap(addr, len)
caddr_t addr;
int len;
```

The `addr` parameter is the value returned from the `mmap` call. The `len` parameter is the length in bytes that you wish to unmap.

It is important to notice that unmapping is independent of the `close` system call. We cannot map a zero length file into memory, plonk 200 bytes of data in the buffer and then `unmap` the new 200 bytes 'back' onto the disk. You have to extend the file somehow. The system call that changes the length of a file is `truncate`. A call to this or `ftruncate` will set the file to the new length.

Using `mmap` to create a random length file is a pain. First you must create the file using the `creat` system call. This will make a zero length file. Next we use the `truncate` call to make the file some 'maximum' size. Now we map this 'maximum' size into memory using `mmap` and add the data into the memory that has been mapped. Let's say we add 3000 bytes. We must now call `truncate` again using 3000 bytes as a parameter to set the file size. Finally, we can unmap the file. Yuck, it's easier to use the standard I/O library.

I am happy to use the `mmap` call only in circumstances where it seems to work well and naturally. I rewrote the accounting code to connect files into memory for processing, and I think this has resulted in a notable speed increase. I do the same for processing the log files which are each around 3 MB. These are mapped into memory, scanned once and unmapped. The performance is good.

Is all this portable? Well, you should be a little circumspect. It's true that not all UNIX systems support virtual memory. It's true that not all UNIX systems supply you with a mmap call at present. It should be coming though.

EXE

Peter Collinson is a freelance consultant specialising in UNIX. He can be reached electronically as pc@hillside.co.uk *(although your mailer might be happier to put the address the other way round) or by phone on 0227 761824.*

MKS RCS When a change . . .

INCLUDES FREE MKS MAKE

. . . isn't as good as the rest

Nobody gets it right first time.

Sometimes you do, but then throw it all away by making so many of those "subtle" little changes, you end up stuck with something that looks nothing at all like how you wanted it to.

Still, that's life. Or at least that's life before MKS RCS - the Revision Control System that keeps a complete history of all the changes made to your files. It lets you retrieve any, or all, of them without any fuss. You can retrieve them by date, by release number, or by whatever name you had previously given them.

Binary or text files? They're handled by MKS RCS with the same contemptuous ease.

Multi-users? No problem. A locking feature prevents more than one person from ringing the changes at any given time.

Branching? You can have as many parallel branches as you like, meaning you really will be able to see the wood for the trees.

Other features? File-compression; a new menu interface; conversion facility from SCCS files; compatible with UNIX system RCS.

Change away to your heart's content. If you *did* get it right first time (or the second, or the third, or the fourth...), you can go back to it at any time you like.

But there's one change you'll want to make permanent. And that's the change to MKS RCS. Call 0763 244144 for details.

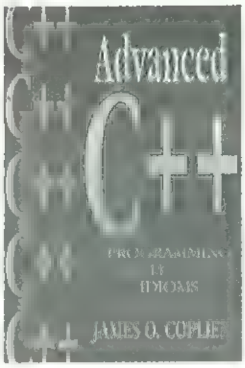
Compatibility: MKS RCS is available for DOS, OS/2, XENIX and 386/UNIX.
For an MKS brochure contact the sole UK distributor:
The Software Construction CompanyLtd, 1 The Maltings, Green Drift, Royston, Hertfordshire SG8 5DB. Telephone: 0763 244114. Fax 0763 244025
MKS RCS, leader in performance AND price: DOS version £165 (5-user licence £645); OS/2 version £225; DOS & OS/2 version £259.

CIRCLE NO. 594

Books

C++ for macho programmers and OS/2 for, well, anybody who's interested...

No laughing matter



Advanced C++ Programming Styles and Idioms. Be warned: do not disregard the title of this book! It does indeed cover some advanced C++ concepts. Even if you are a strong C programmer who is only just beginning to look at C++, you'll want something different.

If, however, you've already learned the semantics and syntax of C++, you'll almost certainly have come across the problems which this book addresses. From an impressive collection of 'names' at AT&T Bell Laboratories, James Coplien has gleaned a series of C++ programming idioms which can benefit us all. These are presented in a clear way, and are followed by guide-lines on when to use them, and the reasons for doing things that way.

The book starts with topics such as data abstraction and inheritance, running through to reuse and dynamic multiple inheritance. Each idiom is discussed in depth, with plenty of useful code examples, and some rather testing problems if you are feeling masochistic (the appendix with the solutions was sadly omitted).

So far, so good. In common with many good works, this has some minor (and some not so minor) irritations. The overall presentation is quirky and haphazard. The appendices seem to contain a miscellany of items for which the author could find no other place (but I could). Some are just long code listings, but others cover important topics such as 'Why bitwise copy doesn't work' and 'Reference return values from operators'.

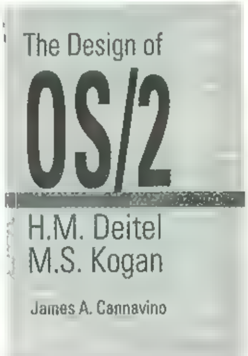
The writing style is quite dry: full of wisdom, but no wit. I don't think the author understands humour, or more relevantly, its importance in striking a rapport with the reader. Worse than this, some of the phraseology is a real block to communication.

Gripes apart, this is an important and useful book. Bjarne Stroustrup gave us a powerful weapon in C++. As he says, C++ may make it more difficult to shoot yourself in the foot, but if you do shoot yourself, you'll blow your leg away. Any well conceived guidance in using such a weapon is surely welcome, but in this case, don't expect it to be fun.

Review by Dr Gareth Blower

Title: Advanced C++ Programming Styles and Idioms *Price: £27.85*
Author: James O. Coplien *Publisher: Addison Wesley*
Pages: 520 *ISBN: 0-201-548-550*

A Recipe for OS/2



The Design of OS/2 by Deitel and Kogan is a slightly misleading title for a book which aims to give an overview of both the 16 and the 32-bit versions of OS/2 (ie OS/2 V1.x and V2.x). They are almost like two different operating systems.

The first part of the book is dedicated to giving the reader an insight into the evolution of the operating system, providing background information on some of the external forces that moulded OS/2. Beginning with the birth of the PC, the authors tell the familiar tale of how IBM asked Microsoft to write the DOS operating system for the new IBM baby. This is followed by a look at the PC hardware. Perhaps the most relevant chapter in the earlier part of the book is the one in which Deitel and Kogan provide a detailed description on how OS/2 was designed to take advantage of memory management on the 286/386.

After the history lesson, *The Design of OS/2* turns its attention towards the architecture of OS/2, covering multi-tasking, memory management and inter-process communication (IPC) on both the 16 and the 32-bit versions of the operating system. The authors look at how processes and threads are controlled under OS/2 and they provide a good argument for the use of multi-threading over a single-threaded operating system. Virtual memory management is examined, but the authors have provided too much detail, making it far more difficult for a casual reader to grasp these concepts. Much of this information belongs in a reference manual. Again, both the 16 and 32-bit APIs are introduced, covering segmented and paged memory management.

Other features of the architecture such as I/O management and Presentation Manager (PM) are also given. I/O management looks at the advantages of the High Performance File System (HPFS) over DOS's FAT and outlines the DEVHELP services.

The Design of OS/2 describes the architectural features of both OS/2 V1.x and OS/2 V2.0. Throughout the book, the authors have described the advantages of OS/2 V2.0 but there is no summary table that list these. It doesn't cover the API in depth but the material that it does present helps to reinforce the theory. My impression is that, if you want to learn more about the OS/2 architecture, *The Design of OS/2* is worth considering.

Title: The Design of OS/2 *Price: £32.25*
Author: H M Deitel and M S Kogan *Publisher: Addison Wesley*
Pages: 389 *ISBN:0-201-54889-5*

Books Received This Month

PC Interrupts by R Brown & J Kyle	Addison Wesley	£27.95	ISBN: 0-201-57797-6	pp1023
Open System LANS by J Houldsworth	Butterworth-Heinemann	£25.99	ISBN: 0-7506-1045-X	pp400
Signal Processing in C by C Reid & T Passin	Wiley	£33.95	ISBN: 0-471-52713-0	pp323

Joining in the Standard

Francis Glassborow, our intrepid C User, spends a day on the Standards panels.

I spent yesterday, 4th February, representing CUG(UK) on two panels of IST; /5/14 - C Panel and /5/21 - C++ Panel. This was the result of our discovery that both panels would welcome contributions from ordinary users. An unusually sensible view for committees sitting on Mount Sinai.

The morning belonged to the C Panel. A standard (ISO 9899) already exists, but one of the conditions for the UK withdrawing its 'no' vote (three no votes at ISO spell failure as a consensus is the objective) was that there would be an addendum refining and clarifying some parts of the Standard. This addendum is rapidly approaching its final stages (two years is fast at this level). Two other countries have complicated the issue asking to make their own contributions to the addendum.

The Japanese have a substantial proposal on multibyte library functions. Even at this late stage a lot of fluidity remains in this proposal. I gather that the underlying principles are acceptable, but getting the details right is taking time. Remember that whatever is finally agreed will need to last at least a decade.

A proposal from, I think, Denmark is much more controversial. They still have many programmers using ISO 646:1983 standard terminals. These lack a number of characters that C programmers expect to use (eg # and {). The trigraph alternatives (??= and ??<) are clumsy and just about unreadable. They want to add other alternatives. These could, theoretically, threaten existing code. The UK vote is in the balance and a 'no' might be critical as a couple of other countries are unhappy with the proposals.

The addendum was by no means the sole business for the meeting and we finally went off to lunch half an hour late, leaving the convener of the C++ Panel getting organised for the afternoon.

Three of the six C Panel members returned after a quick snack at a local hostelry to join the C++ Panel. This meant that 12 of the 14 Panel members were there. The afternoon meeting was in stark contrast to the morning one. This panel is still coming up to speed. The working paper for the Draft Proposed Standard is still slim but much has to be done in getting to grips with the implications before progressively refining the material to an acceptable and viable standard. The excellent work done on the C Standard will help make this task easier. I noticed an impressive depth of experience of writing standards among the panel members.

The main business of the meeting was looking forward to the next meeting of ISO/IEC JTC1/SC22/WG21 and X3J16 (the ISO and ANSI committees working on a C++ standard). This meeting will be held in London, 16th-20th March.

Unlike the C Panel, the C++ Panel adjourned early, but we have plenty of homework and much of it has to be done in the next few weeks.

The final point I would like to make here is that the entire infrastructure (including travel to overseas meetings) for the development and maintenance of standards for computing languages is run from a budget that is less than the earnings of a single consultant.

You will find more details of both the C and C++ panel meetings in the current issue of CVu, the journal of CUG(UK). You will also find details of how to make your contribution in the same place.

EXE

For information about CUG(UK) write to 64 Southfield Road, Oxford, OX4 1PA or ring 0865 246490.

Why the ECUG?

'What, another User Group?' Mike Banahan justifies the European C++ User Group's existence.

Why bother to start a User Group for C++? Mainly because I felt it was nothing like as cut-and-dried as a lot of the other technologies are. C++ has a feel about it which is very much like the early development of UNIX (in the late '70s). They were heady days - people would meet to swap war stories, exchange source code, discuss hints and kinks - and most important of all, get merrily plastered together at technical conferences. Just ask Peter Collinson (you can have the negatives for a fee, Pete).

C++ is at a very similar stage of its evolution. It is clear that it is going to have a substantial effect on the way that people work, even though none of us is sure just what. It's resemblance to C is only that - a resemblance - and the more you learn about it, the more you realise just how different it really is. The tricks and techniques take a long time to learn; people who can reach expert level in simpler languages like C or Pascal within months will tell you that 18 months with C++ leaves you feeling like 2 months into C. Yet everyone agrees that it's worth the effort!

Given all of that, a user group was clearly called for. Our plan is to provide a forum for discussion and conferences, where experience can be spread and ideas tried out on one's peers. The newsletter is already in place and a source-code library is currently being established. We've had a highly successful technical conference in London at the beginning of December, with the next one planned in Munich in the Summer. Already the fragmentary C++ community has something to help to bring it together and we look forward to several exciting years of development before C++ too slumps into the dreary mainstream of commercial programming.

Who is it run by? At the moment an *ad hoc* committee brought together to get it off the ground, comprising a mix of users, academics and vendors of products or services. Once it is properly under way we'll be electing the committee from the membership. It's a not-for-profit organisation, owned by its members. At the moment its main focus is on 'serious' users; almost inevitably this means computing professionals rather than hobbyists, but there is certainly no intention to rule out any of the constituency.

See you in Munich: if you can't get plastered there, you might as well give up altogether ... EXE

Mike Banahan is Chairman of ECUG. Subscription to ECUG is £50 per annum. For more information about its activities, contact Rebecca Thomas on 071 253 5121, or write to ECUG, c/o City House, 190 City Road, London EC1V 2QH.

MARCH .EXEWORD

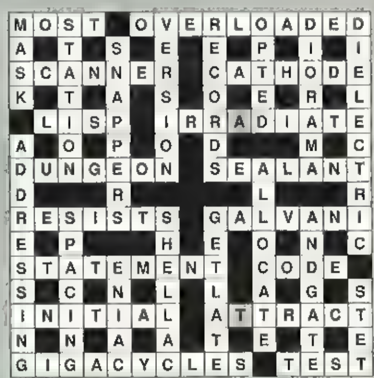
ACROSS

1&3 What our life is all about (4,10)
10 Keep an eye on the screen (7)
11 Embossed work in Italy may provide release (7)
12 Rather cheeky scheduling software (4)
13 The one i/c tapes and discs (9)
15 Disorder growing throughout the Universe (7)
16 Ring with ASCII 7 (4)
18 Trial crucial for new program (4)
21 Old TV films showing what the program does in a loop (7)
24 Set of steps leading to a solution...(9)
25 ...but the program that results may be cryptic (4)
27 In other words, train laziness to keep things going (7)
28 Build the image again (7)
29 Showing initiative is your business (10)
30 "Do it yesterday" in short in CIA's application (4)

DOWN

1 Filing program initially designed by many students (4)
2 Brown fellow puts a graphic line on screen (7)
4 In the country with the first robots, pal (7)
5 Hound with power bug that can be fixed (7)
6 Peaceful greeting in the east (6)
7 List the units of data (7)
8 Unearthed something irrational? (10)
9 Part of network used for training maybe (7)
14 Distribute? Quite the opposite (10)
17 State your requirements clearly (7)
19 Chunk of orange store (7)
20 No festival seems biased (3,4)
21 Asides kept quiet in 25 (7)
22 Where to find the data you want (7)
23 Person needing word processor with all the features (6)
26 IC from the old block? (4)

'.EXEWORD' compiled by Eric Deeson



FEBRUARY .EXEWORD

To advertise in this section, please call Marc Warren on 081-994 6477. Fax 081-994 1533

Give *Real-Time* Projects a Head Start with the AMX™ low-cost high performance Multitasking Kernel

Features

Fast, reliable operation
Compact and ROMable
PC Peripheral support
DOS file access
C and Assembler supported
Preemptive task scheduler
Time slicing available
Configuration Builder
Intertask messages
Message exchanges
Dynamic operations:
- create/delete tasks, timers
- adjust task priorities
- memory allocation
Buffer Manager
Event Manager
Semaphore Manager
List Manager
Breakpoint Manager
Sample and Demo Programs
InSight™ Debug Tool for use with most popular debuggers

Targets

AMX 86 for real mode 80x86
AMX 386 for protected 80386
AMX 68000 for 680x0
AMX 80 for Z80, 64180 and 8085

No royalties
Source code included

For more information and a *free* demo disk contact:

ScotWare Ltd
45 Frederick Street
EDINBURGH
EH2 1EP
Tel: 031-225 4858
Fax: 031-220 3550

Real-time and database software specialists

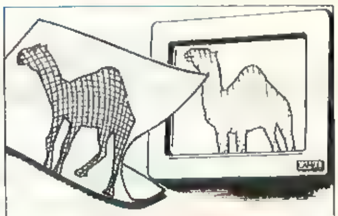
brainy SYSTEMS (UK) LTD

Motherboards

386SX25 £130 w/cache	£150
386DX33 128K Cache	£265
486DX33 256K Cache	£495

.EXE CLASSIFIED LAUNCH OFFER
80486-33 256KB Cache Mini Twr
4MB RAM - 3.5" + 5.25" Floppy
120 MB Hard Disk
2 Serial 1 Parallel 1 Game Ports
Trident 512 KB VGA Card (1MB option)
UK Keyboard (102 Key)
£1299 exc VAT and Delivery

Call Keith Hickson
Tel: 071 734 5783
Fax: 071 734 7256



PLOT THE DIFFERENCE

PLOTVIEW

PLOTVIEW

Phone now for more details about PLOTVIEW the versatile tool that can really make a difference to your productivity.

CAMEL SERVICES LTD.
Telephone OXFORD (0865) 512678

The Secret of Successful Advertising is.....

CLASSIFIED

To find out how you can reach 17,000 professional developers for as little as £85, call Marc Warren on 081 994 6477

windows SHAREWARE

A Different Approach

HD Business Upgrades Custom

ikonWORKS takes a different approach to the distribution shareware for Windows...

HIGH DENSITY DISKS
By compressing on average of 20 programs onto HD disks we offer extremely good value for money.

BUSINESS SOFTWARE
We supply a wide range of applications and utilities which are designed to meet the needs of people who use Windows in business.

FREE UPGRADES
Most of our disks can be returned with your next order for a free upgrade.

CUSTOM DISKS
We can build up disks to order from your choice of over 650 applications, utilities and fonts.

TELEPHONE OR WRITE FOR A **FREE CATALOGUE**

ikonWORKS

KIBWORTH COMPUTER TRAINING

Save time by learning

C or C++

with customised tuition in an ideal environment.

Phone or write for details:

68 Springfield Crescent
Kibworth Beauchamp,
Leicester LE8 0LH

Tel: 0533-792653

Ed *Soft as it Gets*
the programmer's editor

QBS Software Limited,

10 Barley Mow Passage,
London W4 4PH

Tel: 081-994 4842
Fax: 081-994 3441
BBS: 081-747 1979

© Soft As It Gets 1991

Computer MUSIC Systems Ltd

5-7 Buck Street,
London NW1 8NJ.
071-482 5224

***Voyetra* Sound Factory™**

Sound Factory™ is a complete audio development kit for DOS or Windows 3, from Voyetra, the leaders in IBM music software. Containing drivers for a wide range of sound cards, digital audio boards & MIDI interfaces, Sound Factory™ gives developers as wide a market as possible. Call or fax us for further details.

We also stock a wide range of PC software and peripherals.

Specialists in Music Software

BUGS WANTED!

A new BASIC User Group is looking for volunteer regional co-ordinators to help arrange regular meetings around the country.

If you have an interest or any experience with BASIC programming, please write to:

BUG
30 Osprey Close,
West Drayton, Middx UB7 7JE

PC Data Security Guide

Data Protection and Security for Personal Computers is a guide for support staff and managers. Everything you need to know about:

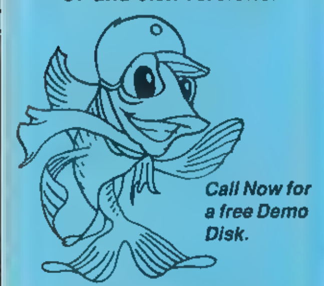
- Keeping viruses out of your company.
- Drafting and implementing a corporate data **security policy.**
- Persuading staff to take regular **backups** that are known to be reliable.
- Detecting and eradicating **software piracy.**

Includes comprehensive product guide and lots of trouble-shooting advice.

The price of £145 includes 90 days' **free telephone support** from the author.

Please call TTK Technical Publications on 081 995 9845 for details and a **free brochure.**

The Babelfish Paradox Data Driver for Clipper Summer '87 and 5.0x versions.



Call Now for a free Demo Disk.

QBS Software Ltd
10 Barley Mow Passage, London W4 4PH
Tel: 081 994 4842
Fax: 081 994 3441

HIRE INFORMATION TECHNOLOGY PLC

H/I/T/E

GUARANTEED VIRUS FREE

Hire I.T. is the **ONLY** PC Rental company that can Guarantee all systems are delivered **Virus Free** and stay that way throughout the hire, because Hire I.T. is the only PC rental company to load Anti-Virus software on every system we ship.

FOR COMPLETE PROTECTION FROM VIRUSES ONLY RENT FROM HIRE I.T.

071 490 1515

OLIVER HOUSE
51/53 CITY ROAD
LONDON EC1Y 2AD

To advertise in this section, please call Marc Warren on 081-994 6477. Fax 081-994 1533

AUTOROUTE Plus

Telephone Version

Simply dial 0839 300100 and use AUTOROUTE !

- Listen to the Service Information
- Use Telephone Key Pad to Select options
- Enter Location Codes from sample list

Aberdeen	00014	Glasgow	02197	Plymouth	[illegible]
Birmingham	00566	Leeds	03192	Reading	04917
Cardiff	01042	London	03424	Southampton	05357
Edinburgh	01804	Manchester	03557	Watford	06052

To obtain a FREE TeleConnection INFORMATION CARD & CODE DIRECTORY Phone 0839 300192 and leave your address details.

Tele Connection

R&R Relational Report Writer

Database reports made easy for dBase, Clipper and compatibles. Also for Paradox, 123s and Quattro Pro.

Turn raw data into useful information using Lotus like commands and menus.

Available from
QBS Software Ltd
10 Barley Mow Passage
London W4 4PH
Tel: 081 994 4842
Fax: 081 994 3441

Call for free demo disk

Fonts & PostScript

PostScript Interpreters: from £4 to £4,000 Fountain, GoScript, Freedom of Press, Ultrascript, PS Tutor, LincPage, NTG2000

PostScript LaserJet Upgrades from £239: Adobe, HP, Pacific Page, UDP TurboScript - HP/Canon

PostScript Utilities: ATM, PSFx, PSPlot, Books

Type 1 Fonts from £3 to £3,000: Public Domain, Adobe, ZSoft, Digi-Duit!, Watland Orb

HP Bitmap Fonts: ZSoft, Typografica, Digi-Duit!, Atech, DTS, Watland Orb, Tailor Made

Laser Printer Memory Upgrades: HP, Canon, IBM

Custom Fonts & Logos: most formats available

Custom HP Font Cartridges: from £90 (10 off)

Orchid '486 & Video

Motherboard + Fahrenheit 1280· runs up to 18 times faster than standard VGA· £960

Complete systems from £2469 with NEC 4FG

David Pollard Associates
Folly Bridge Workshops, Thames Street, Oxford OX1 1SU 0865 240048 Fax 792277

SABREBANNER

For

Helios
Workstation Memory
Sun
Hewlet Packard Apollo
IBM
Apple
Silicon Graphics

Phone: 0532 854521
Fax: 0532 853026

DCL COMPUTERS
COMPUTER PARTS

MOTHERBOARDS

286 AT 20MHz	£115.00
386 SX 20MHz	£250.00
386 AT 25MHz	£453.00
486 AT 25MHz	£1225.00

Inc. IDE, 2H, 2F, 2 serial parallel interface.

HARD DISKS

40Mb HD	£149.00
100Mb HD	£339.00

Other drives available.

For Price List & Details

Tel: 0902-335632
53 Wells Road, Penn,
Wolverhampton WV4 4BQ

Version Control

PVCS*	£345
MKS RCS	£155
SMS	£385
Sourcerers App	£169/399
TLib	£95

Editors

PVCS Editor (SPE)*	£175
QEdit	£45
Brief	£189
Epsilon	£169

Makes

PVCS Make*	£149
MKS Make	£95
Opus Make	£89

Others

SourceDoc*	£199
PolyLibrarian	£175
Demo II	£175
386/BlueMAX*	£69/£85
Project Director*	£495
SegMentor*	£1695

Call for site licences.

Readmar Systems

Tel 071 625 5255

*Distributor ACCESS/VISA

JUST Computer BOOKS

- **Struggling with Sage?**
- **Dicing with DOS 5.0?**
- **Worried by WordPerfect?**

From beginner to expert
We have over 3000 titles to choose from.

Please phone or write for your FREE catalogue

111 Court Rd. Malvern
Worcs. WR14 3EF

☎ 0684-568095

C++ Modeling & Simulation Class Library

Object Oriented Technology for Engineering, Manufacturing, Finance, Education

Meijin++™ 2.0

- Integration & Approximation
- Semi-Persistent Containers
- Discrete Event Simulation
- Optimization & O.R.
- Numerical Analysis
- Queuing Systems
- Exceptions
- Statistics

Royalty Free
115+ Classes
Starting: $349.00
700 Pg User's Manual
MS-DOS/Ws, Unix, Sun
Documented Source Code
Professional Version Available

Network Integrated Services, Inc.
Tel: +1-714-756-0995
221 West Dyer Road
Santa Ana CA 92707
U.S. FAX +1-714-433-2347

IMPRESS YOUR FRIENDS! LOSE THAT FLAB! IMPROVE YOUR SEX APPEAL!

All good ideas - but you won't achieve any of them by

WRITING FOR .EXE

However, unlike jogging, there could be some beer money in it for you. For a copy of our Contributors' Notes, write to: *The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH.*

TAKE UP WEIGHT-LIFTING!

DO YOU NEED MORE MEMORY?

We have three libraries to give you just that!

XMEM - Virtual memory Manager using base, Expanded, Extended or Disk. Replaces malloc. **£145**

MEMORY TOOLKIT - Library of routines to manage Expanded and Extended memory plus big DOS arrays. **£95**

EMPTY SHELL - C routines for freeing up memory when shelling to a large program. Uses EMS, XMS or disk to swap. **£165**

Unit 1m Farrington Fields
Farrington Gurney
Bristol BS18 5UU
Tel 0761 452116
Fax 0761 453226
VAT Extra Carriage Free

Great News!

Object
Oriented
Real
Time
Kernel
for
Turbo Pascal
Now Available
from
JPS Graphics

For more information call
0244 821009
(Telephone or Fax)

JPS Graphics, 3 Glynne Street, Queensferry, DEESIDE, CH5 1TA

JPS Graphics

Clipper PostScript Library
Scripton

There are over 50 functions in Scripton, allowing high definition text and graphics to be produced by PostScript printers. Embed letterheads, logos and graphics in your Clipper systems.

- Improve report presentation
- Eliminate pre-printed notepaper
- Put picture and graphics in reports

Available from: Price: £155.00 + VAT
QBS Software
10 Barley Mow Passage
London W4 4PH
Tel: 081 994 4842 Fax: 081 994 3441

PostScript Utilities

PostVue £99 + VAT
A visual de-bugger for PostScript code. Watch your code execute, examine stacks, dictionaries and state variables. Set break points, warning levels, delay times.

PostPP £49.00 + VAT
Un-scrambles PostScript code and lays it out in human readable form.

PPS £49.00 + VAT
For all PostScript printers. Save paper with this versatile driver for ascii files. Output in columns, landscape, choose fonts, headers, line numbering etc.

Available from
QBS Software
10 Barley Mow Passage
London W4 4PH
Tel: 081 994 4842 Fax: 081 994 3441

PostScript Library for C
Scripton

The successful Clipper library is now available for C programmers. There are over 50 functions in Scripton, allowing high definition text and graphics to be produced by PostScript printers. Embed letter-heads, logos and graphics in your Clipper systems.

- Improve report presentation
- Eliminate pre-printed notepaper

Available from: Price: £155.00 + VAT
QBS Software (Including source)
10 Barley Mow Passage
London W4 4PH
Tel: 081 994 4842 Fax: 081 994 3441

ADVERTISERS INDEX

ADVERTISER	PRODUCT/SERVICE	CIRCLE	PAGE
AI International	OOP Language & Environments	603	58
Applied Logic	Object Oriented Programming	574	55
Automated Office systems	Networking	537	5
Barnet Electronics	PC Hardware	591	79
Biton	Clipper Software	605	92
Bits Per Second I	Graphics for Visual Basic	571	53
Bits Per Second II	Graphics for Clipper	587	75
BL Computer Security	Security Dongle	597	87
Blink	Clipper Fast Linker	586	73
Borland	C++ Video Course	601	27
Brent Communications	MAX copy Protection	549	25
BSO Tasking	Real Time Executive	596	86
Cocking & Drury	SmallTalk V	539	8
Contrast Systems Ltd	SQL Database Systems	556	33
CTL	Copy Protection Hardware	575	55
Custom Business Software	Comms Software	592	80
DES	Software Protection	590	77
Evergreen	CASE Tools	604	29
Expertech	Pro-C	548/563	23
F1 Computing Services	Training	543	17
Facts Direct	FAX Service	569	50
dLesco	Clipper Function Library	599	IBC
G-Force	Graphical Interface for Clipper	572	53
Glockenspiel	Class Constructors	538	7
Grey Matter	Programming Tools	536	3
GWI	Software Design Tools	582	67
Highland Grafix	GUI Software	573	54
Hitex	Embeded Systems	552	28
Inst. Analysts/Programmers	Institute	545	19
Intasoft	Software Management System	558	36
ITEL	Applications Generator	551	26
IXI	Motif	584	69
JPI I	C++ Compilers	544	17
JPI II	C++ Compilers	546	19

ADVERTISER	PRODUCT/SERVICE	CIRCLE	PAGE
Lahey	Fortran Compiler	553	27
LPA	AI/KB/OOPS Software	581	67
Magnifeye	Software Protection Device	589	76
Microcosm	Copy Control	554	27
Microphar	Software Protection	567	46
Microsoft Press	Book Publishers	550	25
Microware	Real Time	564	43
MKS	UNIX Under DOS Tools	594	83
Network Research	Networking	583	67
Next	Workstation	542	14
Nu-Mega	Debugging Tools	541	13
Positive Ltd	Editor	547	20
QA	C Training	535	IFC
QBS	Clipper Function Library	562	89
Readmar Systems Ltd I	Tools	578	19
Readmar Systems Ltd II	Version Control	600	OBC
Salford S/W Marketing	FORTRAN for DOS & UNIX	577	61
Seltec	Software Testing	606	89
Software Con. Co. I	Development Tools	568	49
Software Paradise	Development Tools	579	62/63
Software Security	Security Products	540	11
System C	Application Generator	585	70
System Science	Maths Software	559	37
System Star I	Software Tools	602	25
System Star II	Chart Drawing Software	555	29
System Star III	DOS Extender	561	41
System Star IV	Information Manager	598	41
The Instruction Set	Training	557	35
Thripplewoods	Sanyo Special Offer	595	85
TKH International	Data Compressions	560	38
USA Software	Programming Tools	566	45
User Friendly	Software Copy Protection	588	75
Watcom	Compiler	570	51
Xoren	File Transfer Software	580	64

STOB - A favourite phrase

'One two! One, two! and through and through / The vorpal blade went snicker-snack! / He left it dead, and with its head / He went galumphing back.'

is (of course) bandersnatched from one of the action stanzas of *Jabberwocky*, Lewis Carroll's nonsense poem from (Alice) *Through the Looking Glass*. I thought I would try it out for size, as it were.

To explain: some time ago, an Admirer wrote to me saying, why don't you write a book? Well, I gave it some thought, and the long and short of it is that my *magnum opus*, provisionally entitled *The Art of Computer Programming, Vols 4-10*, is now well under way. But there is much more, I find, to writing a proper computer book than slapping down a few algorithms, banging in an index and faxing the first draft ms through to Messrs Addison-Wiley. Oh yes.

For a start, it's vital to have a soppy dedication at the front, but that's a cinch ('To darling Parity, without whose sisterly support and succour...'). What's brought me up is the little quote that must be provided at the beginning of each chapter. Because you can't just stick in anything you like, you know. There is a strict and heavily policed code - an informal ISO standard, if you like. Any author quoting from, say, the short stories of DH Lawrence risks being laughed out of the BCS. Contrariwise, don't go quoting David Bowie unless you wish to declare yourself a hopeless reverse-snob.

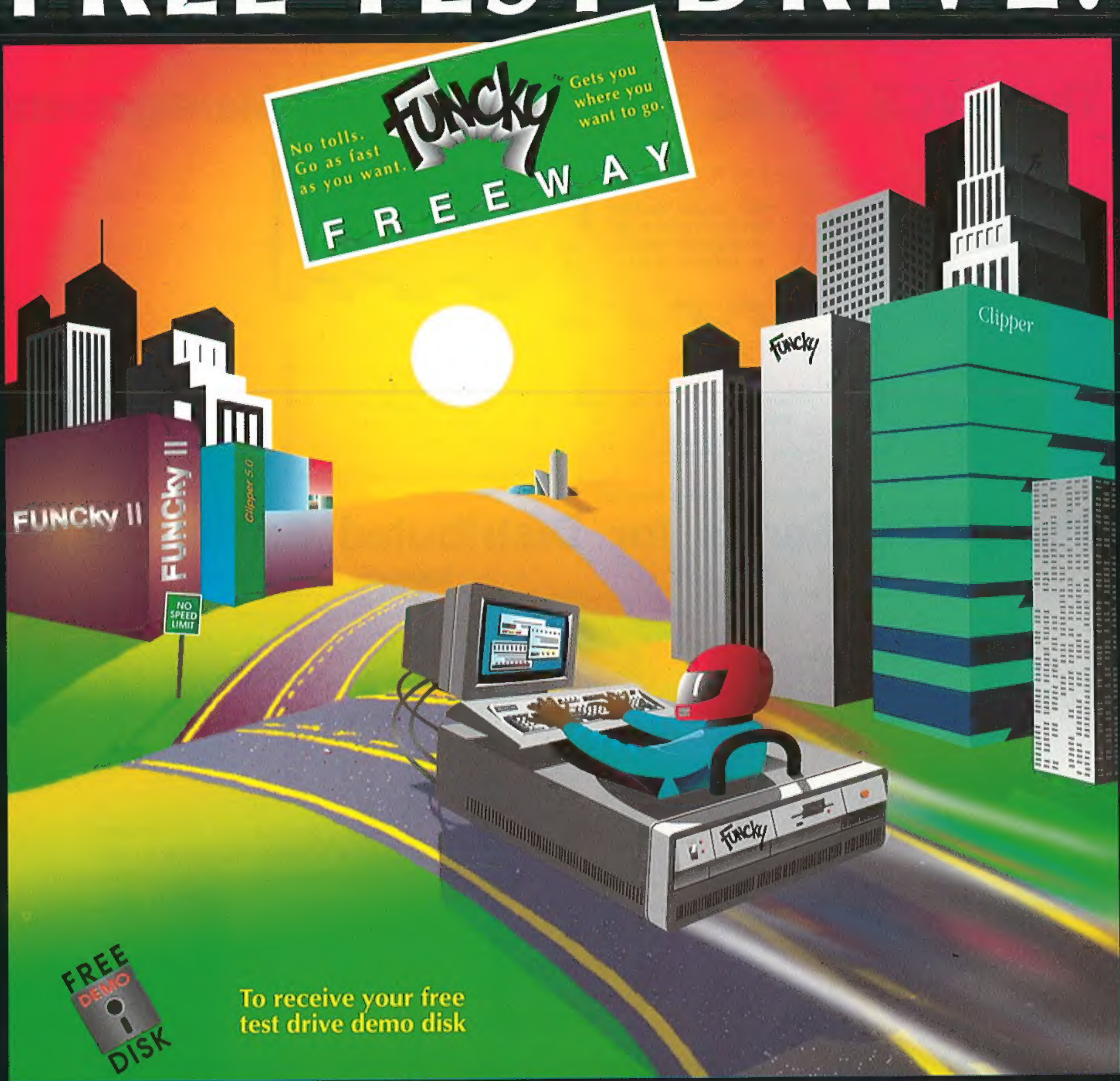
No, a rough, ranked list of acceptable sources for quotations in computer science works is: 1) Lewis Carroll, 2) JRR Tolkien (Hobbitry), 3) The Shakespeare Play you did for O-level, 4) Any colleague at the computer lab, 5) L Frank Baum (Ozery), 6) Somebody who has written a similar text book, 7) Any great scientist from Archimedes through to Rutherford - score double points for obscure ones, 7.5) Countess Ada Lovelace 8) Anybody else, male, and 25) Anybody female.

To see how this works, let's analyse a real book. In Bjarne Stroustrup's *The C++ Programming Language* (2nd Ed) I found 15 chapter heading quotations. I am unable to identify CN Parkinson, Jon Bentley, L Peter Deutsch and F Brooks, better-informed, letter-writing readers may be able to put me straight. For the rest, we have a (slightly mangled?) Henry VI Pt 2, a saying of Bilbo Baggins the Hobbit (at least it wasn't one of those terrible elf-songs which make *Lord of the Rings* the book it is), *two* bits from *Alice*, three quotes of categories 4) and 6), William of Occam (philosopher, mathematician, manufacturer of men's toiletries), W Churchill (huh?), an Einstein and, in Chapter 8 (Templates), a cop-out. It just says 'Your quote here - B Stroustrup'.

Now if a writer of Stroustrup's calibre runs out of original and/or apt quotes, what chance V Stob? I considered inventing a few likely phrases based on what I could remember about Lewis Carroll - ie that he was a rather suspect old gentleman with a penchant for photographing little girls clad only in their birthday suits - but could only come up with 'Togs off, Violet - the camera's ready!', which is of no use to anyone.

Stuff it! I'm going to go my own sweet way, and damn the critics. Chapter 1. The Importance of OOP. *Marilyn Monroe, on being offered matzo balls for the third time by her mother-in-law: 'Isn't there any other part of a matzo you can eat?'* Watch out, Knuth, here I come. .EXE

FREE TEST DRIVE!



Try our innovative TEST DRIVE program, a free demo of the FUNCky library which you can actually program with. Edit code using any or all of the FUNCky library, then compile, link and run to find out for yourself why FUNCky is the ultimate Clipper Developer's tool.

The FUNCky add-in library for Clipper contains over 450 functions which can be used as if they were a part of the Clipper language itself. Extensive features include moveable resizeable windows with mouse support, superior DOS file handling, blazing fast string manipulation, and impressive video control.

Get FUNCky now and get the phenomenal programming power you need to create the advanced applications demanded of you in today's competitive development environment.

dLESKO
INCORPORATED
320 York St., Jersey City, NJ 07302
FAX# 201-435-0637

Contact QBS Software

Telephone: +44 81 994 4842
Fax: +44 81 994 3441
BBS: +44 81 747 1979

CIRCLE NO. 599